THE ULTIMATE BUSINESS TOOL

TAS+

Relational Database

Database/Language
Screen Painter
Program Generator
Report Writer
Runtime Compiler

TAS-PLUS

RELATIONAL DATABASE

COPYRIGHT

Copyright by Business Tools, Inc. 1986. All rights reserved. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means, without the written permission of Business Tools, Inc.

TRADEMARKS

TAS, TAS-Plus, The Accounting Solution and Business Tools are trademarks of Business Tools, Inc. IBM, PC, XT, AT are trademarks of International Business Machines Corporation. dBASE II and dBASE III are trademarks of Ashton-Tate. Wordstar and MailMerge are trademarks of Micropro International Corporation. Microrim, R:BASE SERIES, and CLOUT are trademark of Microrim, Inc. MS-DOS is a trademark of Microsoft, Inc.

DISCLAIMER

This TAS-Plus User Manual is printed in the U.S.A. Any names or references to persons, corporations, or products mentioned in the TAS-Plus User Manual are intentionally fictitious. Any resemblance to persons living or dead or to actual corporations or products is purely coincidental. Business Tools, Inc. believes that the information contained in the manual is correct. However, Business Tools, Inc. does not assume responsibility for the accuracy of the content of the TAS-Plus User Manual nor use of the manual, nor for any patent infringements or other rights of third parties who may use the manual or the software herein. Business Tools, Inc. reserves the right to revise, update or change either the product or the manual without a direct or inferred obligation to Business Tools, Inc. to notify any person of such revisions, updates, or changes.

SOFTWARE COPYRIGHT NOTICE:

This product is only licensed and not sold. It remains the property of Business Tools, Inc. This package contains a written limited warranty, software license agreement, and a sealed system diskette package. If you cannot agree to the software license, return the product to the point of purchase within three (3) days for a full refund, provided all items are returned in the new condition and provided the system diskette package is still sealed.

Unauthorized use of the software or related materials can result in civil damages and criminal penalties.

Acknowledgments

Business Tools, Inc. wishes to acknowledge the contributions made by these people towards the development of TAS-Plus.

PRODUCT DEVELOPMENT

Philip D. Mickelson

DOCUMENTATION

Rick Grafton
Craig G. Ingle
Jennifer S. Rowe

TAS-Plus

Relational Database

Table of Contents

GETTING STARTED

I. What this package contains 2

II. Setting up an IBM floppy disk system 3
- Getting ready 3
- Backups 4
- Making a "bootable" locked System Disk 4
- Adding CONFIG.SYS to your floppy disk 5
- Unlocking your System Disk 6

III. Setting up an IBM hard disk system 7
- Getting ready 7
- Backups 8
- Creating a TAS-Plus Subdirectory 8
- Adding CONFIG.SYS to your hard disk 9
- Unlocking TAS-Plus 10

IV. Specialized setup features 12
- Customizing TAS-Plus default values 12
- Multiple company options 13
- Using TAS-Plus with your printer 13

TUTORIAL

I. Who is TAS-Plus designed for? 14

II. Database fields, records and files 15

III. An introduction to TAS-Plus 17
- File managers, DBMS, and ADES 17
- Data dictionary is the heart 19
- Accessing records with speed 20
- TAS-Plus programming language 21

IV. Creating a quick "Customer" database 22
- Three ways to create a database 22
- Using the "Create database" function 22

V. Maintaining a database 42
- Add a record to a file 45
- Find a record in a file 50
- Change (Edit) a record 54
- Delete a record from a file 58

VI. Browsing a database 60
- Viewing several records at once 61
- Changing records with Browse 63

VII. Creating a new "Sales" program 67
- Design your database on paper first 68
- Editing the data dictionary 72
- TAS-Plus Screen Painter 72
- TAS-Plus Program Generator 89
- Compiling your program with TAS-Plus 91

VIII. Creating a TAS-Plus report 92
Two ways to create a report.....92
Using the "Create Report" function 93
Running a TAS-Plus report.....98

IX. Getting TAS-Plus files to relate 101
Relating your sales and customer files.....102
Using the Source Code Editor.....102
Using your related files.....113

X. Creating a TAS-Plus menu system.....116

XI. Enhancing your Application 125
Accessing system date & time.....125
Adding a password.....131

XII. Advanced TAS-Plus topics 134
Using arrays.....134
If, Then, Else logic.....135
Rules for using braces.....137
Entry options.....139
Date/time arithmetic.....141
Sorting with Indexes.....142
Multiple companies option.....144

REFERENCE GUIDE

I. Conventions in TAS-Plus Programming and Documentatoin 148

II. Schematic of TAS-Plus language 151

III. Main Menu Explained 152

IV. Maintaining Data Dictionary 162

V. Source Code Editor 165

VI. Screen Painter 171

VII. List of TAS-Plus commands 176

VIII. Expressions: Operators & Heirarchy 307

IX. How to convert data types 308

X. List of error messages 310

APPENDICES

I. System requirements 326

II. Technical specifications 327

III. Converting TAS files to non-TAS files 329

IV. Converting non-TAS files to TAS 333

V. Adding color and highlights to TAS programs 337

VI. Installing and Customizing TAS-Plus 340

VII. Function Keys 343

INDEX 347

TAS-PLUS

GETTING STARTED

I. What this package contains

Thank you for purchasing the TAS-Plus Relational Database. We believe TAS-Plus is the finest product of its kind available today.

Before you get started, take a few minutes to make sure you received everything we shipped you from the factory. This package should contain the following:

TAS-Plus Relational Database Manual
Software license Agreement
TAS-Plus Locked Distribution Disk
TAS-Plus Source Distribution Disk

If any of these items are missing, you should immediately contact the dealer you purchased the package from.

If you purchase TAS-Plus directly from Business Tools, Inc., please call Customer Support at 1-206-392-9781, or write Business Tools, Inc., Customer Support, 4038-B 128th Ave. S.E., Suite 266, Bellevue, Washington 98006.

II. Setting up an IBM floppy disk system

TAS-Plus Relational Database is normally recommended to run on a hard disk system. TAS-Plus, however, is actually very compact and will run on a system with two floppy disk drives under certain conditions.

The issue, as with all sophisticated relational databases, is lack of disk storage space on the 360k IBM floppy disks. This will limit both the size of applications you build and the amount of data you can store. For example, if you want to build a simple Mailing List to keep track of 1,000 customers, you can do that by putting the application on one disk and the data on another disk.

If, however, you're planning to build an interactive Sales Entry System to track 10,000 inventory items, 5,000 customers and 2,500 transactions per month, you will need a hard disk system. (NOTE: You could start by building a portion of the application on a floppy disk system, then later upgrade to a hard disk system and complete the application.)

GETTING READY

Before you can begin to use TAS-Plus Relational Database on your IBM (or IBM-compatible) floppy disk system, there are a few things you'll need to do.

First, make sure you have the following.

IBM-compatible computer with two floppy disk drives.
Both original TAS-Plus disks
MS-DOS System Master disk.
Three blank floppy disks.

For more information on specific system requirements, read Appendix I titled System Requirements.

BACKUPS

You should never use your original distribution disks for anything more than just to copy from. The original disks should be kept in a safe place at all times.

TAS-Plus is not copy protected. You don't need any special disk to make it run. You may make as many copies as you need for your personal use. However, as your software license agreement states, you may not sell or give those copies to anyone else for their use.

For more information on copying disks, you should read the MS-DOS manual that came with your computer.

MAKING A "BOOTABLE" LOCKED SYSTEM DISK

Your TAS-Plus Relational Database is divided into two distribution disks - the Locked Disk and the Source Disk. It's important to understand what these disks are and how you use them.

The Locked Disk is the complete TAS-Plus Relational Database. It has a full implementation of the language and all the commands that appear in the manual. You can use it to create, modify and run TAS-Plus applications. In fact, it has only one limitation. The Locked Disk can only store 250 records in a file.

This limited file size makes the locked version of TAS-Plus perfect for building demo applications to see how TAS-Plus works. It's also useful for distributing* your own applications where you don't want the end user to have access to a fully functional version.

> ***NOTE:** Contact Business Tools, Inc. for permission before doing this commercially.

To make a "bootable" Locked TAS-Plus System Disk, simply format a blank disk with MS-DOS system tracks. Then copy your TAS-Plus LOCKED - DISTRIBUTION COPY files onto it.

Here's how to make a "Bootable" Locked TAS-Plus System Disk.

1. Turn on your computer and insert the MS-DOS System disk in drive A.

2. Insert a blank disk in drive B.

3. When you see the A> prompt sign on the screen, type in **FORMAT B: /S**. This will create a "bootable" blank disk.

4. After it's finished formatting, it will ask if you want to format another disk. Type **N** for no.

5. When you see the **A>** prompt sign on the screen, remove the MS-DOS System disk from drive A and replace it with the TAS-Plus LOCKED - DISTRIBUTION COPY.

6. Type in **COPY A:*.* B: /V** and hit the Return key.

7. After it's finished copying, the **A>** will reappear. Remove the disk from drive B and label it TAS-Plus LOCKED - WORKING COPY.

8. Remove the TAS-Plus LOCKED-DISTRIBUTION COPY from drive A and put it away. Place the new LOCKED-WORKING COPY in drive A and reboot the computer.

ADDING CONFIG.SYS TO YOUR FLOPPY DISK

In order for TAS-Plus to make use of all MS-DOS features, it is necessary to add a CONFIG.SYS file to your working copy of TAS-Plus before using it for the first time. If you are using subdirectories, CONFIG.SYS must be in the Root Directory. Here is what to do:

1. Insert your MS-DOS System Disk in drive A and your TAS-Plus working disk in drive B.

2. Check to see if there is already a CONFIG.SYS file on the TAS-Plus working copy. If it is there, use EDLIN or a text editor to add the following lines:

FILES = 40
BUFFERS = 15

3. f there is no CONFIG.SYS file on your working disk, you will need to add one. Here is how to do it:

Type **B:** and press Return Key
Type **COPY CON: CONFIG.SYS**
Press Return Key
Type **FILES=40**
Press Return Key
Type **BUFFERS=15**
Hold down CTRL Key and press **Z**
(You will see **^Z** on the screen)
Press Return Key

UNLOCKING YOUR SYSTEM DISK

You'll find the TAS-Plus Source Disk in a sealed diskette package. Don't open this unless you can accept the enclosed software license agreement.

To remove the 250 record limit from your "bootable" TAS-Plus LOCKED - WORKING COPY follow these steps.

1. Turn on your computer and insert the MS-DOS System disk in drive A.

2. Insert your TAS-Plus LOCKED - WORKING COPY disk in drive B.

3. When you see the **A>** prompt sign on the screen, remove the MS-DOS System disk from drive A and replace it with the TAS-Plus Source Disk.

4. Type in **COPY A: *.EXE B: /V** and hit the Return key. This will update certain files and unlock them.

5. After it is finished copying, the **A>** will reappear. Remove the disk from drive B and label it TAS-Plus SOURCE - WORKING COPY.

III. Setting up an IBM hard disk system

TAS-Plus Relational Database is normally recommended to run on a hard disk system. The only restriction you will face on the size of the applications you can build and the amount of data you can store will be the limitations of your hard disk. To paraphrase the immortal bard, "You can never have too much memory."

GETTING READY

Before you can begin to use TAS-Plus Relational Database on your IBM-compatible hard disk system, there are a few steps you'll need to do.

First, make sure you have the following.

IBM-compatible computer with one hard disk drive and one floppy disk drive.
Both original TAS-Plus disks
MS-DOS installed in the root directory of your hard disk.
Three blank floppy disks.

For more information on specific system requirements, read Appendix I titled System Requirements.

BACKUPS

You should never use your original distribution disks for anything more than just to copy from. The original disks should be kept in a safe place at all times.

TAS-Plus is not copy protected. You don't need any special disk to make it run. You may make as many copies as you need for your personal use. However, as your software license agreement states, you may not sell or give those copies to anyone else for their use.

CREATING A TAS-PLUS SUBDIRECTORY

The best way to work with TAS-Plus on a hard disk system is to put TAS-Plus in its own subdirectory. This subdirectory is simply a section of the hard disk memory that has been set aside exclusively for the use of TAS-Plus Relational Database. It's also easier to make backups by first copying a floppy disk into an empty subdirectory and then back out to a blank floppy. That's what we're going to do now.

Here's how to make a TAS-Plus subdirectory.

1. Turn on your computer without any disk in drive A.

2. When you see the **C>** prompt sign on the screen, type **CD ** this will make sure you are in your Root Directory.

3. Type **MD \TAS** and hit the Return key. This will create a new subdirectory for TAS-Plus.

The next step is to use this empty subdirectory to make a copy of your TAS-Plus Locked Disk. Here's how you do it.

4. Type **CD \TAS** and hit the Return key. This will get you into the new subdirectory called TAS.

5. Insert the TAS-Plus Locked Disk into drive A.

6. Type **COPY A:*.* C: /V** and hit return. This will copy what's on the floppy into the new subdirectory on the hard disk.

7. After it is finished copying, the **C>** will reappear. Remove the TAS-Plus Locked Disk from drive A and store it in a safe place.

8. Insert a blank formatted disk in drive A.

9. Type **COPY C:*.* A: /V** and hit the Return. This will copy what's in the new subdirectory of the hard disk out onto the floppy.

10. After it is finished copying, the **C>** will reappear. Remove the disk from drive A and label it TAS-Plus LOCKED - WORKING COPY.

You now have a working copy of TAS-Plus in its own subdirectory on the hard disk and a backup copy on a floppy disk.

For more information on formatting or copying disks, you should read the MS-DOS manual that came with your computer.

ADDING CONFIG.SYS TO YOUR HARD DISK

If you have a hard disk, TAS-Plus should already be in it's own subdirectory called **TAS**. It is necessary to add a CONFIG.SYS file to your Root Directory (not the TAS subdirectory) before using TAS-Plus for the first time. Here is how:

1. Check to see if there is already a CONFIG.SYS file on the TAS-Plus working copy. If it is there, use EDLIN or a text editor to add the following lines:

FILES = 40
BUFFERS = 15

2. If there is no CONFIG.SYS file in your Root Directory, you will need to add one. Here is how to do it:

Type **C:** and press Return Key
Type **COPY CON: CONFIG.SYS**
Press Return Key
Type **FILES=40**
Press Return Key
Type **BUFFERS=15**
Hold down CTRL Key and press Z
Press Return Key

UNLOCKING TAS-PLUS

Your TAS-Plus Relational Database is divided into two distribution disks - the Locked Disk and the Source Disk. It's important to understand what these disks are and how you use them.

The Locked Disk is the complete TAS-Plus Relational Database. It has a full implementation of the language and all the commands that appear in the manual. You can use it to create, modify and run TAS-Plus applications. In fact, it has only one limitation. The Locked Disk can only store 250 records in a file.

This limited file size makes the locked version of TAS-Plus perfect for building demo applications to see how TAS-Plus works. It's also useful for distributing* your own applications where you don't want the end user to have access to a fully functional version.

> ***NOTE:** Contact Business Tools, Inc. for permission before doing this commercially.

You'll find the TAS-Plus Source Disk in a sealed diskette package. Don't open this unless you can accept the enclosed software license agreement.

To remove the 250 record limit from your "bootable" TAS-Plus LOCKED - WORKING COPY follow these steps.

1. Turn on your computer without a floppy disk in drive A.

2. When you see the **C>** prompt sign on the screen, type **CD \TAS** and hit the Return key. This will get you into the new subdirectory called TAS.

3. Insert your TAS-Plus LOCKED - WORKING COPY disk in drive A.

4. Type in **A: *.EXE C: /V** and hit the Return key. This will update certain files on the hard disk and unlock TAS-Plus.

5. After it is finished copying, the **C>** will reappear. Remove the TAS-Plus LOCKED-WORKING COPY disk and store in a safe place.

You should also make a floppy backup of the unlocked version at this time.

IV. Specialized setup features

CUSTOMIZING TAS-PLUS DEFAULT VALUES

TAS-Plus makes use of your computer's function keys to perform the most common application and database commands. For example, you press F1 (Function key 1) to get on screen Help.

These functions have been preset at the factory so you can use TAS-Plus straight out of the box.

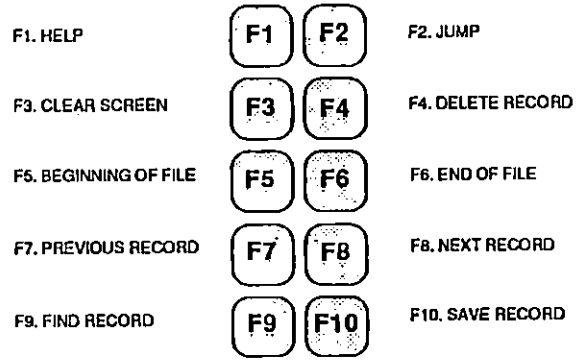


Figure G 1 Function Keys Used by TAS-Plus

You can also redefine these function keys to fit a custom application if need be. The instructions for this are found in Appendix VII,Function Keys.

MULTIPLE COMPANY OPTIONS

One of the unique features of TAS-Plus is its ability to support up to 1,296 sets of data with the same program. With this feature, multiple companies can share the same set of accounting applications. This can be very useful to individuals that handle multiple sets of company books.

For a complete discussion of how to use the multiple companies option, see chapter XII, Advanced TAS-Plus Topics.

USING TAS-PLUS WITH YOUR PRINTER

There are no special requirements for using TAS-Plus with your printer. TAS-Plus will send its output to any parallel or serial printer that is connected to your computer under direction from the MS-DOS operating system.

If you should have any problem printing, check your MS-DOS operator's manual and make sure you have configured your output port correctly. Also make sure you have the printer plugged into the port that you have configured.

TAS-PLUS

TUTORIAL

I. Who is TAS-Plus designed for?

If your job requires you to keep track of information , from simple mail lists to sophisticated order tracking systems, then TAS-Plus is for you.

TAS-Plus can be used as a quick and easy way of storing and retrieving information. Using its "Create a Database" function, you can create data files in minutes - without any programming.

TAS-Plus can also be used to create finished menu driven software applications. This is very different from the simple file manager or even most other relational databases.

If you are just starting out with databases, TAS-Plus will help you get going quickly by providing a one step way of creating a database file. You can store up to 65,000 records in each file and find any of them in just seconds.

TAS-Plus also provides several powerful programs designed to work together to let you be your own application programmer. These include a:

Data Dictionary
Screen Painter
Program Generator
Source Code Editor
Report Writer
Runtime Compiler

Whether you're a novice or a seasoned pro, TAS-Plus will let you create menu-driven business applications more quickly and more easily than you thought possible.

TAS-Plus gives you the maximum flexibility in creating new applications and building on existing applications. At the same time TAS-Plus helps you create database applications that are compact, fast, and powerful.

II. Database Fields, Records, and Files

Let's take a few minutes to talk about what a database is. In one sense, a database is any system used to store information (data). A good example of a simple database is your Index Card File. It's a system for storing names and address on small 3" x 5" cards.

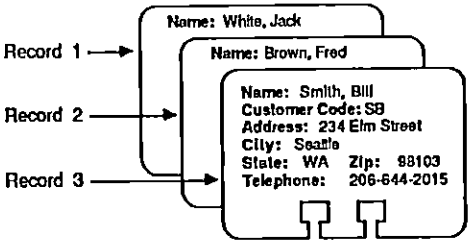


Figure T1 Records are like Index Cards

In a computerized database such as TAS-Plus, you would refer to each index card as a **Record**. For example, Bill Smith would have his own card (or record). Fred Brown would have his own card (or record). And so on.

To add more people to the Index Card File, you simply add more cards. The same is true with a computerized database, you simply add a new record for each new person. TAS-Plus lets you have up to 65,000 records in a file.

In the example of the Index Card File, each card contains similar types of data. That is, each card has a person's name, his address, and so on. With a computerized database like TAS-Plus, each piece of data is called a **Field**. These fields are the basic building blocks of a database.

Typically you give each field a unique name. So in the Index Card File example, there is a Name field, an Address field, a City field, a State field, and a Zip field.

Fields can be named anything you want. However, a field name usually relates to the type of data you plan to store in that field. For example, you wouldn't name a field "color", if you were going to keep data in it like "Seattle," "Atlanta," or "Boise."

TAS-Plus keeps it's records in something called a **File**. This file is the computer equivalent of an office file cabinet. You can give a file any name you wish. But it is best to give it a name that relate to what it contains. This will make it easier for you to remember which file contains which records.

If you used TAS-Plus to computerize your Index Card File, you might name the File "Customer" since it contains the names and addresses of your customers.

Another way to visualize a database is to think of it as a table of rows and columns. In fact, this is the classic manner in which databases are described in computer literature.

Field (Column)

Name	Code	Address	City	State	Zip	Telephone
Brown, Fred	BF	14 East 53rd	Boise	ID	84056	134-783-9876
Smith, Bill	SB	234 Elm St	Seattle	WA	98103	206-644-2015
White, Jack	WJ	54 Manor Pl	Atlanta	GA	43045	299-456-7890

Record (Row)

Figure T2 A File is made up of Records (Rows) and Fields (Columns)

In this example, each horizontal row of the table is considered a record and contains all the data about that person. Each vertical column is considered a field.

Let's say you wanted to find the address of Bill Smith from our previous example. First you would go horizontally down the rows until you found the Smith record. Then you would go vertically across the columns until you found the Address field.

In a sense, a similar type of thing happens when TAS-Plus searches for the Smith address. The difference is TAS-Plus can search thousands of records in the time it takes you to search only a few. This is one of the real strengths of a computerized database like TAS-Plus. It is very fast.

III. An introduction to TAS-Plus

After reading the section above, you might be thinking that all database programs are pretty much the same. Unfortunately that isn't so. There are really three general categories of database programs. Each is designed to do different things.

FILE MANAGERS, DBMS, AND ADES

The simplest form of database program is called the **File Manager**. As the name implies, this type of program is designed to manage a single file at at time.

The advantage of a File Manager is it's fairly easy to learn and use. The disadvantage is it doesn't do very much. File manager programs allow you to add records, delete records and search for records in only one file at a time. Because of this, they typically aren't used for serious business applications.

You could use a file manager to computerize the Index Card File in the above example, since that database contained only one file. Let's say you then wanted to add a second file such as a Sales Journal to your database. At that point a simple File Manager would no longer work. You would need a **DBMS** (Data Base ManagementSystem) to manage two or more files.

A DBMS is more sophisticated than a File Manager and because of this it is usually more difficult to master. However, a DBMS gives you several advantages over a simple file manager. First off, you can have multiple files active at the same time.

Even more important, a DBMS lets you relate files to one another. This is done by selecting certain fields that appear in more than

one file. For example, you might have a Customer File which contained a field called **Custcode** (short for customer code). If the same **Custcode** field also appeared in our Sales File, it would be possible to relate the two files.

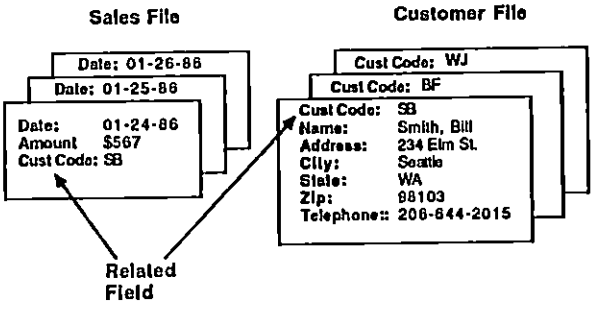


Figure T3 Related Field appears in both the Customer File and the Sales File

Relating files can be very important in creating real world database applications. Let's say you want to send a "valued customer" letter to each person in your Customer File who has made a single purchase of over $500. To do this you will need to search both the files shown above. First you will search the Sales File for records in which the Amount field is over $500. Then when you find one, you will need to find out who the customer code belongs to. You accomplish this by **relating** the record via the **Custcode** to record in the Customer File with a matching **Custcode**.

The next step up from the DBMS is the **ADE** or Application Development Environment. ADEs are programs used to create other programs (applications). Typically ADEs include a powerful relational database plus a way of creating custom menus, data input screens and reports.

In some ways an ADE is similar to a DBMS. For instance, both incorporate relational databases. The difference lies in how each is used. The DBMS is typically used by one person to keep track of his or her personal or company data. In other words, the person creating the database is the same person who will be using the database.

Storing the specifications in a Data Dictionay greatly increases the ease and speed with which you can create multi-file applications. For example, let's say you wanted to build a Customer File that would relate to a Sales File using a common **Custcode** field. You would have to make certain that Custcode used exactly the same specifications in each file or the files would not be able to relate.

Now imagine you are building a sophisticate accounting applications with 40 or 50 individual little programs - many of which contain **Custcode** fields. Now imagine there are several hundred fields like **Custcode** also making up this application. If even one of these fields contained a mistake in the field specification, the entire accounting application would fail to function properly.

TAS-Plus solves this problem by verifying that named fields always use the same specifications, no matter which program they appear in.

ACCESSING RECORDS WITH SPEED

Another benefit of working with TAS-Plus is speed. With TAS-Plus you can typically find any one of 65,000 records in a database in under three seconds. With smaller databases, finding a record can be almost instantaneous.

This is possible because every record is sorted at the moment it is orignially saved. In fact, up to 16 different sorts can be maintained on-line for each database. With TAS-Plus, you never have to resort a database just to find a record.

TAS-Plus uses a system known as a B+Tree. A B+Tree has two things that make it faster than other database systems. First, it doesn't have to actually sort the records, just small parts called **Keys**. These **Keys** are like name-tags that TAS-Plus uses to identify each record.

Second, TAS-Plus searchs for records in a special way known as binary searching. This method allows TAS-Plus to find a record

The ADE, on the other hand, is typically used by one person to create an application which will be used by someone else. For example, a business owner or consultant might use an ADE to create a specialized Order Tracking System that would be used by the company's sales staff.

The level of knowledge required to create the finished Order Tracking System would be much greater than that needed to actually use it. By using the ADE, one person has made his knowledge available to others in the form of an easy to use program.

TAS-Plus is a very special program since it can be used as an ADE, as a DBMS or as a File Manager. It combines a powerful DBMS with a programming language and an easy-to-use interface."

If you are new to databases, you can start out using TAS-Plus as a File Manager. As your knowledge and requirements grow, you will find that TAS-Plus is a powerful DBMS. When you are ready to share that knowledge with others by developing your own applications, you will see why we think TAS-Plus is the best ADE available today.

This ability to serve many different functions and to create other business tools (software applications) is why we say TAS-Plus is the Ultimate Business Tool.

DATA DICTIONARY IS THE HEART

One of the reasons TAS-Plus is so flexible and powerful is that it is built around a Data Dictionary. The Data Dictionary is a special TAS-Plus data file that is used to keep track of every specification of every field in every record and file you create.

This means that once you define a field you never have to worry about defining again. Every time you want to use that field in a different database or program, TAS-Plus automatically looks up the specifications in the Data Dictionary.

without having to search the through the whole database. In fact, it only has to look at six records to find the one in 65,000.

If this all sounds a bit technical, don't worry about it. You don't have to know how a B+Tree works to benefit from using it. TAS-Plus takes care of all the details for you.

TAS-PLUS PROGRAMMING LANGUAGE

TAS-Plus is one of those rare programs that seems to grow as your understanding grows. A big part of this is due to the unique TAS-Plus programming language that is included with each database.

The TAS-Plus programming language is a powerful structured 4th Generation Language. It includes over 86 commands and 200 options. Many of the TAS-Plus commands can be compared to macro-functions found in other high level languages. In other words, a single TAS-Plus command can often replace several entire lines of Basic code.

The TAS-Plus language also allows you to directly access non-TAS programs, thus greatly expanding what can be accomplished within a TAS-Plus program.

So much for our discussion of how databases work. Now let's take a look at how TAS-Plus puts this all into practice.

IV. Creating a quick "Customer" database

If you're like most people, you probably can't wait to try out your new software package. That's no problem, since you can start using TAS-Plus without knowing anything about programming.

THREE WAYS TO CREATE A DATABASE

There are three ways to create a new database with TAS-Plus. The first way is to use the Data Dictionary to define the fields that will make up the file. The second way is to use the **Create Program** function which is part of the **Program** option on the main menu. This function creates both a database and a program to go with it. The third way is to use the **Create database** option.

The **Create database** function is the quickest way to get started with TAS-Plus. It allows you to create single file databases just by answering questions on the screen. You can then immediately start entering data .

In this section we'll show you step-by-step how to create a sample database to keep track of your customers. You will learn how to use the **Create database** function which is part of the **Database** option on the TAS-Plus Main Menu.

USING THE "CREATE DATABASE" OPTION

An important feature of the **Create database** function is that it does a lot of the work for you. For example, as you answer questions about each field, it automatically updates your data dictionary for you. This means that if you decide to use the same fields in another database or program later on, TAS-Plus will "remember" the specifications for you.

In this example you are going to create a database to keep track of the following information about your customers:

1. Code
2. Name
3. Company
4. Address
5. City
6. State
7. Zip
8. Area Code (Telephone)
9. Exchange (Telephone)
10. Phone Number (Telephone)
11. YTD Purchases
12. Date of Last Order

You will use the **Create database** option to create a database with these12 data fields. When you are finished you will have a working database. You can then add records to it. You can also change, delete or search for records.

Since this is your first time using TAS-Plus, let's start at the very beginning.

> **Note:** This section assumes that if you have a floppy disk system, you have all ready made a backup and a bootable working copy of your TAS-Plus disk. If you have a hard disk system it assumes that you have already created a subdirectory called **\TAS** and copied TAS-Plus into it

A. If you are using a floppy disk system, put your bootable working copy of TAS-Plus in disk drive A . Turn on the power switch for both the computer and the monitor.

B. If you are using a hard disk system, turn on both your computer and monitor. When you see **C>** on the screen, type **CD \TAS** and press the return key.

C. Next, with either a floppy or a hard disk system, type **TAS** and press the return key.

---

1. The first step is to get the TAS-Plus Main Menu on the screen.

It should look like this:

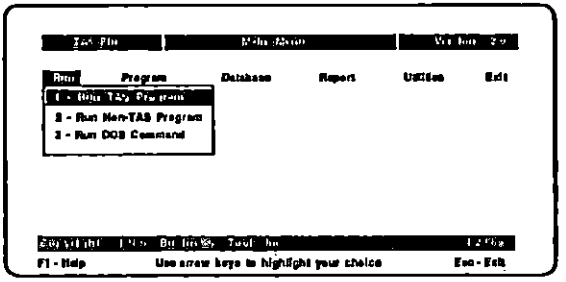


Figure T4 - Initial Main Menu

2. Use your left/right arrow keys to move to the **Database** option. Notice that as you move from one option to another, a different window appears under each. This shows the functions available by choosing that particular option. Under the Database option are three choices:

Create database
Maintain database
Browse database

3. Use your up/down arrow keys to choose **Create database**, then press the return key.

4. The screen should now look like this:

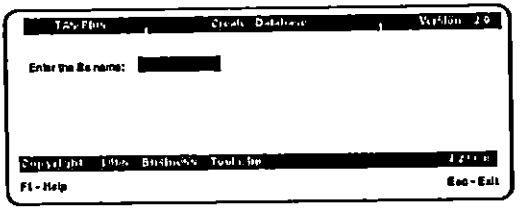


Figure T4A - Initial Create Database Screen

5. The screen will ask you to "Enter the File name:_____." Type in **CUSTOMER**, then press the return key.

6. You should now see a blank **Enter a New Field** screen like the one shown below:

```
Enter a New Field

Field Name:          ################
File Name:           ########
Field Type:          #
     Size:           ###
     Dec Chrs:       #
Upper Case (Y/N):    #
Key (Y/N):           #
Number of elements
     in array:       ###
```

7. The cursor will be in the top field titled "Field Name." Type in **CUSTCODE**, then press the return key.

Note: The field titled "File Name" will automatically be filled in as **CUSTOMER**.

8. A window will appear in the middle of the screen showing the choices available for "Field Type."

```
Enter a New Field

Field Name:          CUSTCODE
File Name:           CUSTOMER
Field Type:          A
        --Fld Types----------
        | A - alphanumeric  |
        | N - numeric       |
        | S - short date    |
        | D - long date     |
        | T - time          |
        ---------------------
```

Since **A - alphanumeric** is already highlighted, simply press the return key to choose it.

9. Next the screen will ask you for the "Field Size." Type **3**, then press the return key. Your screen should look like this.

```
Enter a New Field

Field Name:          CUSTCODE
File Name:           CUSTOMER
Field Type:          A
      Size:          3
      Dec Chrs:      #
Upper Case (Y/N):    #
Key (Y/N):           #
Number of elements
      in array:      ###

Maximum size of alpha field is 255 characters
```

Note: The screen will automatically jump over the "Dec Chrs:" field. This is because the field has been defined as an alpha instead of a numeric.

10. The screen now asks if you only want to allow upper case letters in the field. The default is **Y**, so just press the return key to accept it. Your screen should now look like this.

```
Enter a New Field

Field Name:          CUSTCODE
File Name:           CUSTOMER
Field Type:          A
     Size:           3
     Dec Chrs:
Upper Case (Y/N):    Y
Key (Y/N):           #
Number of elements
     in array:       ###

Are all entries to be upper case only? (Y/N)
```

11. Next the screen will ask if the field is to be a Key. Type **Y** for yes, then press the return key. Your screen should now look like this.

```
Enter a New Field

Field Name:          CUSTCODE
File Name:           CUSTOMER
Field Type:          A
     Size:           3
     Dec Chrs:
Upper Case (Y/N):    Y
Key (Y/N):           Y
Number of elements
     in array:       ###

Are all entries to be upper case only? (Y/N)
```

Remember that you can only search for and report by fields that are defined as keys.

Note: The screen will automatically jump over the "Number of elements in array" field since key fields can not be arrays.

12. The screen will then ask "Is all of the above correct?" If everything is the way you want it, just press the return key to accept the default of yes. The screen should look like this.

```
Enter a New Field

Field Name:              CUSTCODE
File Name:               CUSTOMER
Field Type:              A
     Size:               3
     Dec Chrs:
Upper Case (Y/N):        Y
Key (Y/N):               Y
Number of elements
     in array:

Is all of the above correct (Y/N) Y
```

Note: If you made an error or if you want to change something, type N for no and you will go back to the top of the screen.

13. Two things happen at this point. First, the specifications for the field you just enter will appear on a line toward the bottom of the screen. Each time you enter a new field, it will be shown this way.

```
-----------------------------------------------------
|              Enter a New Field                    |
|                                                   |
|     Field Name:                                   |
|     File Name:                                    |
|     Field Type:                                   |
|          Size:                                    |
|          Dec Chrs:                                |
|     Upper Case (Y/N):                             |
|     Key (Y/N):                                    |
|     Number of elements                            |
|          in array:                                |
|                                                   |
|  Add more fields?  (Y/N)                          |
-----------------------------------------------------
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTCODE	CUSTOMER	A	3		Y	Y	

14. Second, the screen will ask if you want to "Add more fields?" The default is **Y** for yes, just press the return key to go on.

15. At this point you have completely defined one field. The screen will again show you a blank **Enter a New Field** form and wait for you to define the next field.

16. Enter the following specifications for the **CUSTNAME** field (which is the next field) just like you did for the previous one. When you are finished your screen will look like this.

```
-----------------------------------------------------
|          Enter a New Field                        |
|                                                   |
|   Field Name:          CUSTNAME                   |
|   File Name:           CUSTOMER                   |
|   Field Type:          A                          |
|        Size:           25                         |
|      Dec Chrs:                                    |
|   Upper Case (Y/N):    N                          |
|   Key (Y/N):           Y                          |
|   Number of elements                              |
|        in array:                                  |
|                                                   |
|  Is all of the above correct?  (Y/N)      Y       |
-----------------------------------------------------
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTCODE	CUSTOMER	A	3		Y	Y	

17. Enter the following specifications for the **CUSTCOMP** field (which is the next field) just like you did for the previous one. Since this is not a key field, the screen will ask you for **Number of elements in array**. Just press the return key to skip over this question. When you are finished your screen will look like this.

```
Enter a New Field

Field Name:              CUSTCOMP
File Name:               CUSTOMER
Field Type:              A
     Size:               25
     Dec Chrs:
Upper Case (Y/N):        N
Key (Y/N):               N
Number of elements
     in array:

Is all of the above correct? (Y/N)          Y
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTCODE	CUSTOMER	A	3	Y	Y		
CUSTNAME	CUSTOMER	A	25	N	Y		

18. Enter the following specifications for the **CUSTADDR** field (which is the next field) just like you did for the previous one. Since this is not a key field, the screen will ask you for **Number of elements in array.** Press the return key to skip over this question. When you are finished your screen will look like this.

```
+-------------------------------------------------------------+
|                    Enter a New Field                        |
|                                                             |
|        Field Name:              CUSTADDR                    |
|        File Name:               CUSTOMER                    |
|        Field Type:              A                           |
|              Size:              25                          |
|              Dec Chrs:                                      |
|        Upper Case (Y/N):        N                           |
|        Key (Y/N):               N                           |
|        Number of elements                                   |
|              in array:                                      |
|                                                             |
|   Is all of the above correct?  (Y/N)          Y            |
+-------------------------------------------------------------+
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTCODE	CUSTOMER	A	3		Y	Y	
CUSTNAME	CUSTOMER	A	25		N	Y	
CUSTCOMP	CUSTOMER	A	25		N	N	

19. Enter the following specifications for the **CUSTCITY** field (which is the next field) just like you did for the previous one. Since this is not a key field, the screen will ask you for **Number of elements in array**. Press the return key to skip over this question. When you are finished your screen will look like this.

```
Enter a New Field

Field Name:              CUSTCITY
File Name:               CUSTOMER
Field Type:              A
     Size:               25
     Dec Chrs:
Upper Case (Y/N):        N
Key (Y/N):               N
Number of elements
     in array:

Is all of the above correct?  (Y/N)        Y
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTCODE	CUSTOMER	A	3	Y	Y		
CUSTNAME	CUSTOMER	A	25	N	Y		
CUSTCOMP	CUSTOMER	A	25	N	N		
CUSTADDR	CUSTOMER	A	25	N	N		

20. Enter the following specifications for the **CUSTSTATE** field (which is the next field) just like you did for the previous one. Since this is not a key field, the screen will ask you for **Number of elements in array.** Press the return key to skip over this question. When you are finished your screen will look like this.

> **Note:** Only the last five lines of field information are displayed at the bottom of the screen. As you enter more fields, the old information will scroll off. The fields, of course, are not lost.

```
Enter a New Field

Field Name:              CUSTSTATE
File Name:               CUSTOMER
Field Type:              A
     Size:               2
     Dec Chrs:
Upper Case (Y/N):        Y
Key (Y/N):               N
Number of elements
     in array:

Is all of the above correct?  (Y/N)      Y
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTCODE	CUSTOMER	A	3		Y		
CUSTNAME	CUSTOMER	A	25		N		
CUSTCOMP	CUSTOMER	A	25		N		
CUSTADDR	CUSTOMER	A	25		N		
CUSTCITY	CUSTOMER	A	25		N		

21. Enter the following specifications for the **CUSTZIP** field (which is the next field) just like you did for the previous one. When you are finished your screen will look like this.

```
Enter a New Field

Field Name:              CUSTZIP
File Name:               CUSTOMER
Field Type:              A
     Size:               5
     Dec Chrs:
Upper Case (Y/N):        Y
Key (Y/N):               Y
Number of elements
     in array:

Is all of the above correct? (Y/N)          Y
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTNAME	CUSTOMER	A	25	N	Y		
CUSTCOMP	CUSTOMER	A	25	N	N		
CUSTADDR	CUSTOMER	A	25	N	N		
CUSTCITY	CUSTOMER	A	25	N	N		
CUSTSTATE	CUSTOMER	A	2	Y	N		

22. Enter the following specifications for the **CUSTAREA** field (which is the next field) just like you did for the previous one. Since this is not a key field, the screen will ask you for **Number of elements in array**. Press the return key to skip over this question. When you are finished your screen will look like this.

```
Enter a New Field

Field Name:              CUSTAREA
File Name:               CUSTOMER
Field Type:              A
     Size:               3
     Dec Chrs:
Upper Case (Y/N):        Y
Key (Y/N):               N
Number of elements
     in array:

Is all of the above correct?  (Y/N)        Y
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTCOMP	CUSTOMER	A	25	N	N		
CUSTADDR	CUSTOMER	A	25	N	N		
CUSTCITY	CUSTOMER	A	25	N	N		
CUSTSTATE	CUSTOMER	A	2	Y	N		
CUSTZIP	CUSTOMER	A	5	Y	Y		

23. Enter the following specifications for the **CUSTEXCH** field (which is the next field) just like you did for the previous one. Since this is not a key field, the screen will ask you for **Number of elements in array**. Press the return key to skip over this question. When you are finished your screen will look like this.

```
Enter a New Field

Field Name:              CUSTEXCH
File Name:               CUSTOMER
Field Type:              A
     Size:               3
     Dec Chrs:
Upper Case (Y/N):        Y
Key (Y/N):               N
Number of elements
     in array:

Is all of the above correct?  (Y/N)        Y
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTADDR	CUSTOMER	A	25	N	N		
CUSTCITY	CUSTOMER	A	25	N	N		
CUSTSTATE	CUSTOMER	A	2	Y	N		
CUSTZIP	CUSTOMER	A	5	Y	Y		
CUSTAREA	CUSTOMER	A	3	Y	N		

24. Enter the following specifications for the **CUSTPHNUM** field (which is the next field) just like you did for the previous one. Since this is not a key field, the screen will ask you for **Number of elements in array**. Press the return key to skip over this question. When you are finished your screen will look like this.

```
-------------------------------------------------------------
|              Enter a New Field                            |
|                                                           |
|    Field Name:              CUSTPHNUM                     |
|    File Name:               CUSTOMER                      |
|    Field Type:              A                             |
|         Size:               4                             |
|         Dec Chrs:                                         |
|    Upper Case (Y/N):        Y                             |
|    Key (Y/N):               N                             |
|    Number of elements                                     |
|         in array:                                         |
|                                                           |
|  Is all of the above correct?  (Y/N)          Y           |
-------------------------------------------------------------
```

Field Name	File Name		Type	Size	Dec	Up Case	Key	Array
CUSTCITY	CUSTOMER	A	25		N	N		
CUSTSTATE	CUSTOMER	A	2		Y	N		
CUSTZIP	CUSTOMER	A	5		Y	Y		
CUSTAREA	CUSTOMER	A	3		Y	N		
CUSTEXCH	CUSTOMER	A	3		Y	N		

25. Enter the following specifications for the **CUSTYTD** field (which is the next field) just like you did for the previous one. Since this field is numeric, the screen will ask you for **Number of Decimal Characters**, type **2** for two characters after the decimal point. When you are finished your screen will look like this.

```
-------------------------------------------------------------
|                    Enter a New Field                      |
|                                                           |
|    Field Name:              CUSTYTD                       |
|    File Name:               CUSTOMER                      |
|    Field Type:              N                             |
|         Size:               10                            |
|         Dec Chrs:           2                             |
|    Upper Case (Y/N):        N                             |
|    Key (Y/N):               Y                             |
|    Number of elements                                     |
|         in array:                                         |
|                                                           |
|  Is all of the above correct?  (Y/N)          Y           |
-------------------------------------------------------------
```

Field Name	File Name		Type	Size	Dec	Up Case	Key	Array
CUSTSTATE	CUSTOMER	A	2		Y	N		
CUSTZIP	CUSTOMER	A	5		Y	Y		
CUSTAREA	CUSTOMER	A	3		Y	N		
CUSTEXCH	CUSTOMER	A	3		Y	N		
CUSTPHNUM	CUSTOMER	A	4		Y	N		

26. Enter the following specifications for the **CUSTLASTORD** field (which is the next field) just like you did for the previous one. Since you are defining this as a date (short form), the field size will automatically be set to 8. Also, since this is not a key field, the screen will ask you for **Number of elements in array.** Press the return key to skip this question. When you are finished your screen will look like this.

> **Note:** This is the final field to be entered. Remember to type **N** for no when the screen asks if you want to add more fields.

```
        Enter a New Field

  Field Name:          CUSTLASTORD
  File Name:           CUSTOMER
  Field Type:          S
       Size:           8
       Dec Chrs:
  Upper Case (Y/N):    N
  Key (Y/N):           N
  Number of elements
       in array:

Is all of the above correct?  (Y/N)      Y
```

Field Name	File Name	Type	Size	Dec	Up Case	Key	Array
CUSTZIP	CUSTOMER	A	5		Y		Y
CUSTAREA	CUSTOMER	A	3		Y		N
CUSTEXCH	CUSTOMER	A	3		Y		N
CUSTPHNUM	CUSTOMER	A	4		Y		N
CUSTYTD	CUSTOMER	N	10	2	N		Y

27. After you type **N** for no more fields to add, you will see the records all flash by quickly on the screen. This is just TAS-Plus checking to see that everything is all right. After a few moments the screen will display the following message:

```
|                                                          |
| Now please wait while I initialize the file for use      |
------------------------------------------------------------
```

27. At this point, TAS-Plus will automatically create a database for you based on the field specifications you just typed in.

28. It only takes TAS-Plus a few moments to initialize (create) the database for you. As soon as it is finished, the screen will show you the following message:

```
    ------------------------------------------------------
    | Do you want to add records now? (Y/N) Y |
    ------------------------------------------------------
|                                                          |
| Now please wait while I initialize the file for use      |
------------------------------------------------------------
```

29. Congratulations. You have just created you first TAS-Plus database. At this point you can quit by typing **N** for no and you will be returned to the TAS-Plus Main Menu.

If you want start working with your **CUSTOMER** database right away, just type **Y** for yes and go on to Chapter 5, Maintaining a database.

V. Maintaining a database

Now that you have created a **CUSTOMER** database with TAS-Plus, you can start using it right away. You can add records, change records, delete records and generally keep things up to date. In computerese, this type of activity is known as "maintaining a database."

> **Note:** This chapter assumes that you have already created the **CUSTOMER** database. If you have not, then go back to Chapter 4 and begin there.

If you are coming directly from Chapter 4 you can skip over the next part and go directly to step number 3.

If you exited the program after Chapter 4, then do the following steps.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Database** option.

2. Use your up/down arrow keys to choose **Maintain database** and press the return key.

3. The screen will ask you to "Enter the File Name:____."

Note: If you are coming directly from chapter 4, then TAS-Plus will already have entered **CUSTOMER** as the file name for you.

At this point you can type in the name of the file you want or you can press **F2** (Function key 2) and TAS-Plus will give you a list of all the files available on the system.

4. To see how this works, go ahead and press **F2**.

A window will appear in the middle of the screen with a list of all active files. It should look like this:

```
-----------------------------------
|  A  -  CUSTOMER                 |
|  B  -  TASDICT                  |
|  C  -  TASFILE                  |
|  D  -  TASOTHER                 |
|  E  -  TSINGLE                  |
-----------------------------------
```

The up/down arrow keys let you move from one choice to another. Try this to see how it works.

> **Note:** We are actually only interested in choice **A - CUSTOMER** which is the database file we created in chapter . The others are files that TAS-Plus uses internally.

5. Use the up/down arrow keys to choose **A - CUSTOMER** and press the return key. TAS-Plus will automatically enter **CUSTOMER** as the file name.

6. The screen will then ask you to "Enter the fields to be displayed? (All/Selected).

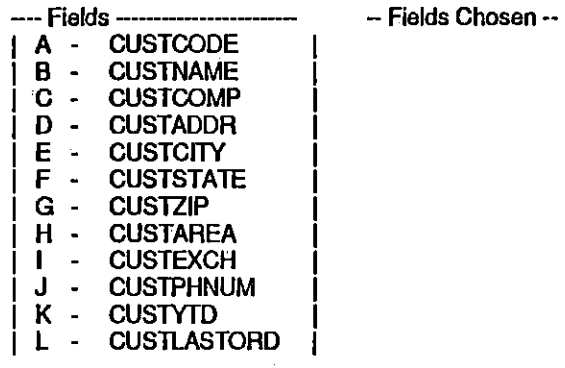
> **Note:** Typically you would use the All choice when entering new records into a database. The Select choice is good for updating just certain fields of records.

7. At this point you can type **A** and press the return key.

If you would like to see which fields are in the file, you can press **F2** (Function key 2) and TAS-Plus will give you a list of all the fields. This is not something we need to do for this example, however it is quick and easy way to find out what fields are in a file. It's very handy when using the Select choice.

8. Just to see how this works, go ahead and press **F2**.

A window will appear in the middle of the screen with a list of all fields in the file.



Again, you can use the up/down arrow keys to move from one choice to another. We don't need to use this function for our example so we won't actually select any fields. However, if we were going to select fields, it would be a simple matter of highlighting them and pressing the return key. They would then appear on the right hand side of the screen under "Fields Chosen."

> **Note:** If curiosity gets the better of you and you choose some fields above, just press the **Esc** key to get back to a blank screen and start over.

Now let's get back to our example. We left off in step six with the screen asking you to "Enter the fields to be displayed ? (All/Selected) S."

9. Type **A** for all and press the return key.

You are now ready to begin entering records into your **CUSTOMER** database.

ADD A RECORD TO A FILE

The first thing you need to do with any new database is add some records to it. In this section you will learn how to input data into the **CUSTOMER** database.

First let's take a tour of the screen. Notice that TAS-Plus displays a complete list of all the fields down the left side of the screen. Notice also that some of these have > signs preceding them. This is TAS-Plus' way of reminding you these are key fields.

To the right of the field names is a series of reversed video areas known as data input blocks. This is where you type in the data for each field. The screen should look like this:

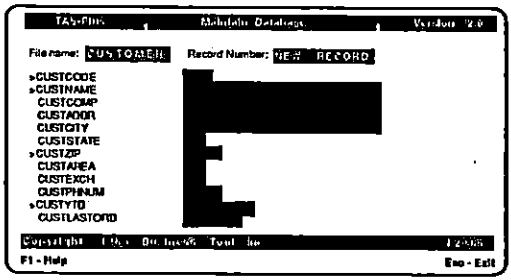


Figure T3 - Initial Customer Entry Screen

OK, let's get started.

1. The screen is now waiting for you to enter the customer's code in the **CUSTCODE** data input block. Type **001** and press the return key.

> **Note:** With Alphanumeric fields the cursor always starts on the left-most position of the data input block. With Numeric fields, the cursor starts at the decimal point.

2. Next you will enter the customer's name in the **CUSTNAME** data input block. Type **Alhabin, Abdul** and press the return key.

3. Next you will enter the company name in the **CUSTCOMP** data input block. Type **Middle Eastern Imports** and press the press the return key.

4. Next you will enter the customer's address in the **CUSTADDR** data input block. Type **33 Main Street** and press the press the return key.

5. Next you will enter the city in the **CUSTCITY** data input block. Type **Lincoln** and press the press the return key.

6. Next you will enter the state in the **CUSTSTATE** data input block. Type **IA** and press the press the return key.

7. Next you will enter the zip code in the **CUSTZIP** data input block. Type **56001** and press the press the return key.

8. The next three data input blocks are used to record the customer's complete phone number. It also would be possible to use a single field to record a phone number, however, a later program in this manual will require the number in this format.

9. You will enter the telephone area code in the **CUSTAREA** data input block. Type **431** and press the return key.

10. Next you will enter the first three digits of the telephone number in the **CUSTEXCH** data input block. Type **467** and press the return key.

11. Next you will enter the last four digits of the telephone number in the **CUSTPHNUM** data input block. Type **9834** and press the return key.

12. Next you will enter the dollar amount purchased year-to-date in the **CUSTYTD** data input block. Type **125.43** and press the return key.

> **Note:** The cursor will start at the decimal point. Numbers will be enter to the left until you type a period and numbers will be entered to the right of the decimal point.

13. Next you will enter the date of the last order in the **CUSTLASTORD** data input block. Type **011386** and press the return key.

> **Note:** Date fields will automatically be displayed with slash marks separating the month, day and year. You only need to type in the actual numbers.

14. This completes the data entry of the first record. The input area of your screen should look like this:

File name: **CUSTOMER** Record Number: **NEW RECORD**

```
> CUSTCODE    001
> CUSTNAME    Alhabin, Abdul
  CUSCOMP     Middle Eastern Imports
  CUSTADDR    33 Main Street
  CUSTCITY    Lincoln
  CUSTSTATE   IA
> CUSTZIP     56001
  CUSTAREA    431
  CUSTEXCH    467
  CUSTPHNUM   9834
  CUSTYTD     125.43
  CUSTLASTORD 01/13/86
```

Save this record Y/N? Y

15. The screen will now ask if you want to "Save record? Y/N Y." The default is **Y** for yes. If everything is correct, press the return key to accept **Y**.

> **Note:** If you have made a mistake or just want to change something, answer **N** for no and you will go back to the first field.

16. At this point TAS-Plus adds the record to your database and saves the data to disk. It then clears the screen so you can enter another record. If you wanted to quit, you can do so now by pressing the **Esc** key. You would return to the TAS-Plus Main Menu.

17. You should get some more experience adding records, so we have listed some sample data below for you to enter.

These records will be used in later portions of this manual, so it is important that you enter them now.

18. Enter the following data for record number two.

```
> CUSTCODE    002
> CUSTNAME    William, Ralph V.
  CUSCOMP     Pro Musica
  CUSTADDR    202 Maple Lane
  CUSTCITY    Bridgeport
  CUSTSTATE   CT
> CUSTZIP     01255
  CUSTAREA    217
  CUSTEXCH    444
  CUSTPHNUM   4321
  CUSTYTD     1376.35
  CUSTLASTORD 02/18/86
```

19. Enter the following data for record number three.

```
> CUSTCODE    003
> CUSTNAME    Larson, David
  CUSCOMP     Little Greenhouse
  CUSTADDR    702 Monroe Avenue
  CUSTCITY    Monroe
  CUSTSTATE   WA
> CUSTZIP     98720
  CUSTAREA    206
  CUSTEXCH    234
  CUSTPHNUM   8765
  CUSTYTD     150.45
  CUSTLASTORD 02/01/86
```

20. Enter the following data for record number four.

```
> CUSTCODE    004
> CUSTNAME    Manrique, Rick
  CUSCOMP     La Villa Restaurant
  CUSTADDR    1774 East Washington Ave.
  CUSTCITY    Santa Monica
  CUSTSTATE   CA
> CUSTZIP     91543
  CUSTAREA    213
  CUSTEXCH    765
  CUSTPHNUM   2576
  CUSTYTD     2584.64
  CUSTLASTORD 03/15/86
```

21. Enter the following data for record number five.

```
> CUSTCODE    005
> CUSTNAME    Levins, Brenda
  CUSCOMP     Bedroom Boutique
  CUSTADDR    15543 NE Webster Street
  CUSTCITY    Chicago
  CUSTSTATE   IL
> CUSTZIP     45903
  CUSTAREA    519
  CUSTEXCH    567
  CUSTPHNUM   4313
  CUSTYTD     346.86
  CUSTLASTORD 01/03/86
```

22. Enter the following data for record number six.

```
> CUSTCODE    006
> CUSTNAME    Reed MD, Frank
  CUSCOMP     Santa Rosa Sports Clinic
  CUSTADDR    1568 Fourth Ave
  CUSTCITY    Santa Rosa
  CUSTSTATE   CA
> CUSTZIP     95478
  CUSTAREA    707
  CUSTEXCH    432
  CUSTPHNUM   5854
  CUSTYTD     2376.76
  CUSTLASTORD 03/01/86
```

That's all there is to adding a record with TAS-Plus. You can press the **Esc** key to exit to the TAS-Plus Main Menu or you can go on to the next section.

FIND A RECORD IN A FILE

Now that you have created a CUSTOMER database and saved some records in it, you can begin experimenting with some of the features of TAS-Plus.

As mentioned earlier in the manual, TAS-Plus can typically find any one of 65,000 records in less than three seconds. TAS-Plus does this by searching for records according to key fields. You will recall that when you designed the CUSTOMER database, you designated three fields to be keys. They were:

> CUSTCODE
> CUSTNAME
> CUSTZIP

Note: You can only search by key fields.

This means you can search for records in the CUSTOMER database by customer code, by customer name or by zip code. You could also start at the beginning or the end of the file and step through the records one by one. Of course that would take a lot more time than searching for a key.

> **Note:** This chapter assumes that you have already created the **CUSTOMER** database and added records to it. If you have not, then go back to Chapter 4 and begin there.

If you are coming directly from the **ADD A RECORD** section, then you can skip over the next part and go directly to step 5.

If you exited the program after the last section, then start with step one below.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Database** option.

2. Use your up/down arrow keys to choose **Maintain database** and press the return key.

3. The screen will ask you to "Enter the File Name:____."

Note: At this point you could press **F2** (Function key 2) and TAS-Plus will give you a list of all the files available on the system.

4. Type **CUSTOMER,** then press the return key.

5. The screen will then ask you to "Enter the fields to be displayed? (All/Selected).

> **Note:** At this point you could press **F2** (Function key 2) and TAS-Plus will give you a list of all the fields that make up this file.

6. Type **A** for all, then press the return key.

7. The screen should now show you the file name as **CUSTOMER** and the record number as **NEW RECORD**.

The screen will show you a complete list of all the fields down the left side of the screen. Notice that some of these have > signs preceding them. This is TAS-Plus' way of reminding you these are key fields.

To the right of the field names is a series of reversed video areas known as data input blocks. This is where you type in the data for each field. The screen should look like this:

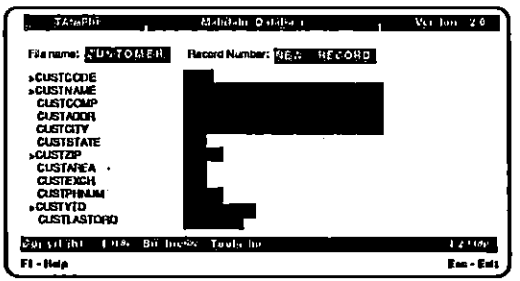


Figure T8 - Initial Customer Entry Screen

8. Use your up/down arrow keys to move the cursor to the **CUSTNAME** data input block.

Now let's say you want to find the record for Brenda Levins. Here is what to do.

9. With the cursor still showing in the **CUSTNAME** data input block. Type **Levins, Brenda.**

10. Next press the **F9** (Function Key 9) which is the Find Record key. The complete record will appear on the screen.

What you have just done is known as a "complete field" search. In other words, you typed in the complete field of characters which you wanted to search for, then TAS-Plus went into the database and hunted for an exact match.

It is also possible to do a "partial field" search when you don't know exactly what you are looking for. Let's say you wanted to find the record for a customer named Reed, but you didn't know what his first name was. TAS-Plus will let you do that.

11. First clear the screen of data by pressing **F3** (Function key 3) which is the Clear Screen key. All the information in the data input blocks should disappear.

12. With the cursor still showing the **CUSTNAME** data input block, type **Reed.**

13. Next press the **F9** (Function Key 9) which is the Find Record key. The complete record will appear on the screen.

You can search for records by placing the cursor in any data input block marked as a key field, typing in a partial field or an exact field and then pressing the F9 key. Finding records with TAS-Plus is as simple as that.

While we are talking about finding records, it would be a good time to talk about some other ways of moving around in the database. In addition to F9 - Find Record, there are four other function keys that let you move through your records. They are:

F5 - Find Beginning of File
F6 - Find End of File
F7 - Find Previous Record
F8 - Find Next Record

These function keys work much like the F9 - Find Record key. For example, let's say you wanted to find the first record in the **CUSTOMER** file. First you would clear the screen by pressing the F3 - Clear Screen key. Then you would move the cursor to the **CUSTNAME** data input block. Next you would press the F5 - Find Beginning of File key. The record that appears on the screen will be the first record sorted on that key field. In this case it would be the "Alhabin, Abdul" record since Alhabin's name starts with the letter A.

You could move to the next record (sorted alphabetically by name) simply by pressing the F8 - Find Next Record key.

The F6 - Find End of File and the F7 - Find Previous Record keys work in a similar manner.

> **Note:** You can not search past the beginning or end of a file. For example, if you are already on the last record in a file and you press the F8 - Find Next Record key , you will get an error message.

That's all there is to finding a record with TAS-Plus. You can press the Esc key to exit to the TAS-Plus Main Menu or you can go on to the next section.

CHANGE (EDIT) A RECORD

What happens when one of your customers moves? How do you change his (or her) address? It's really very easy. TAS-Plus lets you change (edit) or add to data that already exists in your database. In this section we'll show you how.

> **Note:** This section assumes that you have already created the CUSTOMER database and added records to it. If you have not, then go back to Chapter 4 and begin there.

If you are coming directly from the **FIND A RECORD** section, then you can skip over the next part and go directly to step 5.

If you exited the program after the last section, then start with step one below.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Database** option.

2. Use your up/down arrow keys to choose **Maintain database** and press the return key.

3. The screen will ask you to "Enter the File Name:____."

   Note: At this point you could press **F2** (Function key 2) and TAS-Plus will give you a list of all the files available on the system.

4. Type **CUSTOMER,** then press the return key.

5. The screen will then ask you to "Enter the fields to be displayed? (All/Selected).

   Note: At this point you could press **F2** (Function key 2) and TAS-Plus will give you a list of all the fields that make up this file.

6. Type **A** for all, then press the return key.

7. The screen should now show you the file name as **CUSTOMER** and the record number as **NEW RECORD.**

The screen will show you a complete list of all the fields down the left side of the screen. Notice that some of these have > signs preceding them. This is TAS-Plus' way of reminding you these are key fields.

To the right of the field names is a series of reversed video areas known as data input blocks. This is where you type in the data for each field. The screen should look like this:

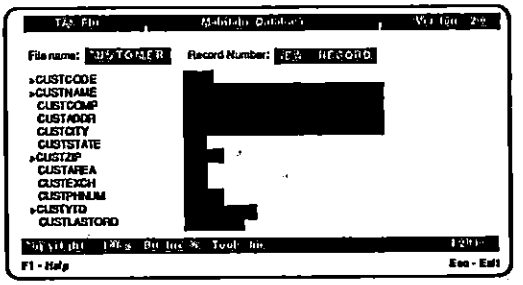


Figure 17 - Initial Customer Entry Screen

8. Use your up/down arrow keys to move the cursor to the **CUSTNAME** data input block.

Now let's say you want to find the record for Brenda Levins. Here is what to do.

9. With the cursor still showing in the **CUSTNAME** data input block. Type **Levins, Brenda.**

10. Press the **F9** (Function key 9). The screen will display the complete record for you.

11. Use the down arrow key to move to the **CUSTADDR** data input block.

Let's say that she moved from Webster Street to 456 Bondman Avenue West. To change the data all you simply type the new address right over the old one. Let's try it.

12. With the cursor setting on the left-most position of the data input block, type **456 Bondman Avenue West**.

> **Note:** If the new data was shorter than the old data, you would just type some spaces to get rid of the old text. Another way of handling this step would be to clear the field first before entering the new data. You can clear a single field by positioning the cursor anywhere within that field's data input block, then holding down the Ctrl (Control) key and typing **U**.

13. Use the down arrow key to move to the last field at the bottom of the screen.

14. While the cursor is still in the data input block for the last field, press the return key.

> **Note:** You can also save the record at this point by pressing **F10** (Function key 10) which is the Save Record key.

15. The screen will now ask if you want to "Save record? Y/N Y."

16. If everything is the way you want, press the return key to accept the default of Y for yes. TAS-Plus will then save the record with the the new address change.

> **Note:** You can change (edit) a record as many times as you want. However, once you have changed a record and saved it, the original is gone forever.

That's all there is to changing a record with TAS-Plus. You can press the **Esc** key to exit to the TAS-Plus Main Menu or you can go on to the next section.

DELETE A RECORD FROM A FILE

Deleting a record from a file works much the same way as changing a record. That is, first you find the record with the F9 - Find a Record key, then you delete it with the F4 - Delete a Record key.

1. The first thing you have to do is find the record that you want to change. We will use the F9 - Find Record key that you learned about in the last section.

2. Use your up/down arrow keys to move the cursor from one data input block to another. Move the cursor to the **CUSTNAME** data input block.

Now let's say you just learned that Brenda Levins sold her business and you want to delete her record from the database. Here is what to do.

3. With the cursor still showing in the **CUSTNAME** data input block. Type **Levins, Brenda**.

4. Press the **F4** (Function key 4).

5. A message will appear at the bottom of the screen asking if you want to "Delete record ? Y/N Y."

6. If you are sure this is the correct record to delete just press the return key to accept the default **Y** for yes.

Let's stop for a moment and take a look at what you have learned so far. You now know how to:

Create a TAS-Plus database
Add records to a TAS-Plus database
Find records in a TAS-Plus database
Update records in a TAS-Plus database
Delete records from a TAS-Plus database

Of course TAS-Plus can help you do a lot more than this, but as of now you know enough about TAS-Plus to start using it to keep track of your own information. Congratulations.

VI. Browsing a Database

In the last two chapters you learned how to work with records one at a time. When you searched for a record, TAS-Plus would go into the database, find that individual record and display it on the screen.

Sometimes though, it would be more convenient to see several records on the screen at a once. TAS-Plus allows you to do this with the **Browse database** function, which is part of the **Database** option on the TAS-Plus Main Menu.

Using the **Browse database** function, you can display 10 records on screen at a time and then scroll through the entire database file. With **Browse database** the records will appear in table format (rows and columns) in any key field order you like. You can display all the fields that make up the records or select just certain ones to display.

> **Note:** If you select all fields, the file will display sequentially in record number order. If you choose to display certain fields, you will be given a choice of presenting them in order of a key or record number order.

In addition to scrolling up and down (from record to record), you can also scroll left and right (from field to field).

The **Browse database** command does a lot more than just let you view records. It also lets you change them. In fact, you could think of **Browse database** as sort of a mini interactive screen report. It lets you "see" the relationship of a record to its neighbors.

Here's an example of how you might use browse. One of the problems that often faces a database users is hunting for mistakes or inconsistencies in a field from one record to another. For instance, when you were typing in the data to the **CUSTOMER** file, you might have typed the "city" information in the "address "field." This type of error is easy to spot with **Browse database.** When you find an error, you can fix it on the spot.

VIEWING SEVERAL RECORDS AT ONCE

So much for the explanation, here's how it works.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Database** option.

2. Use your up/down arrow keys to choose **Browse database** and press the return key.

3. The screen will ask you to "Enter the File Name:___."

   Note: At this point you could press **F2** (Function key 2) and TAS-Plus will give you a list of all the files available on the system.

4. Type **CUSTOMER** and press return.

5. The screen will then ask you to "Enter the fields to be displayed? (All/Selected)."

   Note: At this point you could press F2 (Function key 2) and TAS-Plus will give you a list of all the fields that make up this file.

6. Type **A** (to display all fields) and press return.

7. The first 10 records in the **CUSTOMER** file will display on the screen. The top line shows the filename and the record number.

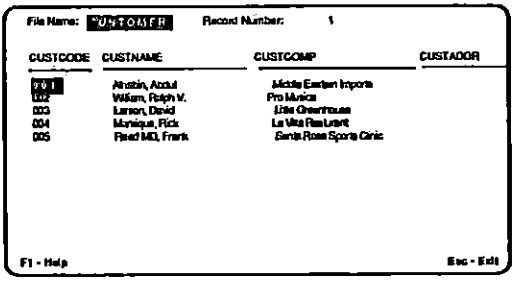


Figure 7B - Browse Screen with Customer Records

When you choose all fields to display, the data is presented in record number order. The first field of the first record is highlighted.

> **Note:** Record number order is not necessarily chronological order. When you first enter records into a file, the records are stored sequentially. When you delete records from a file, TAS-Plus does not actually delete the data. Instead it flags the record as deleted by putting a special mark at the beginning of the record. This way TAS-Plus knows that the record has been deleted and skips over it when accessing a file. After a record has been flagged as deleted, the next time you save a record it is saved over the deleted record instead of at the end of the file. This prevents the file from getting progressively larger and cluttered with unwanted records.
>
> For example, you'll see on the screen that there is no record for customer code 005. That was the record for Brenda Levins which you deleted in the last chapter .

8. Press the **down arrow** key three times to move the highlighted bar down to record 004.

As the bar moves down through the records in the customer code field, watch the record number counter on the top of the screen change. It now shows record number four.

9. Press the **down arrow** one more time.

You'll see the record number counter skip over number five and show number six. This is because number five was deleted. Later if you add another record to the file, the new record will occupy position number five. While you are displaying records by number, it will be displayed between records four and six.

10. Notice that the CUSTADDR (customer address) column on the right side of the screen looks empty. This is because the field is too wide to be displayed on the screen. To display the CUSTADDR you will need to scroll horizontally to the right.

11. Hold down the **Ctrl** (Control) key and press the **right arrow** three times.

The highlighted bar will move across the screen to the right. The third time it is pressed the CUSTCODE and CUSTNAME columns will scroll off the screen to the left and the CUSTADDR and CUSTCITY columns will appear on the right.

CHANGING RECORDS WITH BROWSE

1. Let's say you want to change the address field for Reed MD, Frank, who is custcode 006. You edit the data in **Browse database** mode simply by moving to the field you want to edit, typing over the old information and saving it.

2. Press the **right arrow** key (this time without holding down control). This will move the cursor through the address field. Type in a new address of **1445 Fifth Avenue West** and press **return**. The highlighted bar will move to the next field.

3. Press the **down arrow** to move to the next record.

4. The screen will say "The record has been changed. Do you want to save it? (Y/N) Y." Type **Y** and press the return key to save the record.

> **Note:** Two things must happen in order to make a change permanent. One, you must press the return key after making a change. Two, you must change records by pressing the up or down arrow key.
>
> When you have made changes to a record, you will be prompted to save the record only when leaving the record. You can make as many changes to a record as you want before saving it. If you type **N** at the save prompt, the record will be left as it was.

5. Press **Esc** to leave the **Browse database** function. You will return to the TAS-Plus Main Menu.

6. Now let's take a look at the database in a different way. Let's find out which customers are placing orders and how much they are ordering. The way we'll do this is by using the **Browse database** function and sorting it on the CUSTNAME field.

7. From the TAS-Plus Main Menu, move to the **Database** option. Then select the **Browse database** function.

8. The screen will ask you to "Enter the File Name:___."

9. Type **CUSTOMER** and press return.

10. The screen will then ask you to "Enter the fields to be displayed? (All/Selected)."

11. Type **S** (to select fields) and press return.

12. Type **B** and CUSTNAME will appear under Fields Chosen.

13. Type **K** and CUSTYTD will appear under Fields Chosen.

14. Type **L** and CUSTLASTORD will appear under Fields Chosen.

15. Press **Esc** to signal TAS-Plus you are done with field selection.

16. A new "Keys" window replaces the "Field" window. This one lets you pick the order in which the records will be displayed on screen. The window will look like this:

```
Enter the fields to be displayed? (All/Selected) S

---Keys ----------------------      Fields Chosen
| A - CUSTCODE          |          CUSTNAME
| B - CUSTNAME          |          CUSTYTD
| C - CUSTZIP           |          CUSTLASTORD
| D - Record Number     |
-------------------------------
```

Note: You can choose to display records by any of the Keys listed. This includes CUSTNAME (customer name) even though it is not one of the fields selected for viewing.

The opportunity to select order only occurs when you have selected particular fields to display.

17. Type **B** to choose the CUSTNAME (customer name) as the key.

18. The screen the asks you to "Enter the beginning key value (if any)." Press the return key to select all records in the file.

Note: If you only wanted to look at certain records, you could specify a starting point here.

19. The file will display on the screen according to customer name. Also, only the fields you selected will display. The columns will be in the order you selected the fields. The rows will be in order of customer name.

20. You can experiment with the cursor movement if you like. The following keys allow you to move around through your file:

up arrow	previous record
down arrow	next record
ctrl-right arrow	next field
ctrl-left arrow	previous field
right arrow	next character
left arrow	previous character

21. When you are finished, press the **Esc** key to return to the TAS-Plus Main Menu.

22. That's all there is to viewing and changing records using the **Browse database** function.

VII. Creating a new "Sales" Program

Up until now you have been using TAS-Plus as a simple database. You have created your own **CUSTOMER** database and you have worked with the records in it. All that is well and good. In fact you could accomplish a lot with just what you've learned so far, but TAS-Plus allows you to do much more.

TAS-Plus let's you create complete applications, not just simple databases. Let's take a moment to talk about what an applications is. In its most basic form an application is a program combined with at least one customized data input screen and one database file.

The real significance of this is the customized data input screen. TAS-Plus let's you "paint" your computer screen to look like an actual paper form that you use in the office. Because the computer screen will look like a real life form, it makes it a lot easier for non-computer people to use the applications you create.

There are two ways of creating a new database application with TAS-Plus. One way is to choose the **Create Program** function which is part of the **Program** option on the TAS-Plus Main Menu. This is an all-in-one application generator. It is the fastest and easiest way to create a single-file application.

The second way to build an application is to choose the **Edit Program** function which is also part of the **Program** option on the TAS-Plus Main Menu. This allows you to go directly to the built-in Source Code Editor/Checker. You would use this method to build sophisticated applications, such as accounting programs. You could also use this method to modify or add to existing applications.

> **Note:** TAS-Plus fully integrates its Screen Painter, Source Code Editor and Runtime Compiler under the **Create Program** function. Because of this, it's usually quicker to start all new applications here - even sophisticated ones that will need extensive modification later on.

For our purposes, we'll stay with the **Create Program** function for now. The Source Code Editor will be covered in a later chapter.

> **Note:** This chapter assumes that you have already worked through chapter 4 "Creating a quick Customer database" and chapter 5 "Maintaining a database" and are familiar with the conventions presented there. If not, you should go back and work through those chapters first.

In this chapter you will learn how to use the **Create Program** function to build an application called **SALES**. This **SALES** application is really a simple sales journal. You could use it to record daily sales activity including Order Number, Customer Code, Sales Amount and other pertinent information. In the following chapters, we'll learn how to add a menu and report to this application.

DESIGN YOUR DATABASE ON PAPER FIRST

The first and most important step in building a database application is to design it on paper before even turning on the computer. A little time spent thinking things through now can greatly reduce the time (and frustration) spent fixing things later on.

Your design doesn't have to be fancy . In fact, a good place to start is to list the types of data (information) that will make up an application. Here's a list of what we want to keep track of in our **SALES** example.

1 - Order Number
2 - Date of Transaction
3 - Sales Amount
4 - G/L Account Number
5 - Customer Name
6 - Customer Code
7 - Company Name
8 - Customer Address
9 - Customer City
10 - Customer State
11 - Customer Zip

These items will become the fields that make up each record.

The next thing is to decide how much room will be needed to store data in each field. Also, will the data in each field be letters, numbers, dates, times, etc.

TAS-Plus can keep track of the following data types:

A - alphanumeric
N - numeric
D - long date format
O - overlay
R - record number field
S - short date format
T - time

In our **SALES** example we'll be using A, N, and S (which are the most commonly used data types).

TAS-Plus doesn't care what type of data you store. However, the type you designate will determine what can be done with the data later on. For example, if you expect to add two fields together and put the result in a third field, then all three fields would have to be numeric (numbers) types.

The size of the field is determined by the largest entry you expect to put in that field. For example, you wouldn't make the Customer Name field only five characters long, since most customer names are longer than five characters.

Here are the data types and lengths for the **SALES** example:

	Data	Type	Length
1 -	Order Number	A	4
2 -	Date of Transaction	S	8
3 -	Sales Amount	N	9
4 -	G/L Account Number	A	3
5 -	Customer Name	A	25
6 -	Customer Code	A	3
7 -	Company Name	A	25
8 -	Customer Address	A	25
9 -	Customer City	A	25
10 -	Customer State	A	2
11 -	Customer Zip	A	5

The next thing you have to think about is how you are going to retrieve data from your database. Here's how it works. In order to search for data stored in a specific field or to retrieve data sorted on a specific field, you must designate that field as a **Key** when the application is first set up.

For example, let's say you are going to have a "Moonlight Madness" sale at the end of the month and you want to send a sale letter to all your recent customers. To do that you need to know which customers purchased something during the past month. In other words, you need a list of customers sorted by Customer Name field.

To be able do this, you will need to designate the Customer Name field as a Key when the application is first built. Of course, you can come back later and change things like field type, length and key. But it is a lot easier to plan ahead and do it now.

> **Note:** It would be possible to make all fields keys but this is not a good idea for several reasons. First, you are limited to 16 key fields per database. Second, keys eat up extra memory. Third, the more keys you use, the slower an application runs. A rule of thumb in programming is the fewer the keys the better.

Here are the keys, data types and lengths for the **SALES** example:

	Data	Type	Length	Key
1 -	Order Number	A	4	Y
2 -	Date of Transaction	S	8	N
3 -	Sales Amount	N	9	N
4 -	G/L Account Number	A	3	N
5 -	Customer Name	A	25	Y
6 -	Customer Code	A	3	Y
7 -	Company Name	A	25	N
8 -	Customer Address	A	25	N
9 -	Customer City	A	25	N
10 -	Customer State	A	2	N
11 -	Customer Zip	A	5	Y

The final thing you have to design on paper is what you want the computer screen to look like. Often you can get good ideas by looking at the paper forms around your office. For example, if you were building an invoice tracking system, you might design the computer screen so it looked like your paper invoices. With TAS-Plus you have the ability to make the screen look any way you want.

For our **SALES** example, we'll keep things fairly simple. The data input form will look something like this:

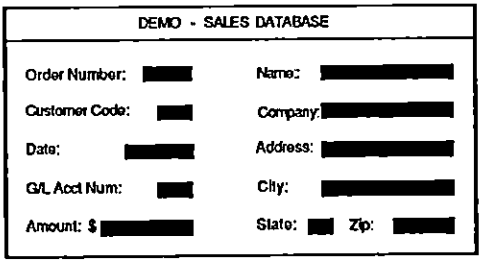


Figure T9 - Sales Input Screen

Don't worry about how you're going to get the screen to look like the above illustration. We'll go through this first example step-by-step.

EDITING THE DATA DICTIONARY

As we discussed earlier in Section III, TAS-Plus uses a Data Dictionary to keep track of the specifications (type, length, key, etc.) of each field that you designate. It is possible to add to or edit the Data Dictionary directly and that will be covered in a later section.

For our **SALES** example though, the Data Dictionary will be edited automatically during the screen "painting" phase of the **Create Program** function. In other words, as you answer the questions (that appear on screen) about each field, the information will automatically be entered into the Data Dictionary for you.

One of the real benefits of TAS-Plus is that you can build a single file application simply by "painting" the screen and answering the questions about each field. TAS-Plus will then write the program for you. In doing so it will also edit your Data Dictionary.

TAS-PLUS SCREEN PAINTER

Now that you know what you are going to keep track of in your database, you are ready to use the Screen Painter to "paint" a data entry form on the screen.

The Screen Painter is part of the **Create Program** functions which is under the **Program** option on the TAS-Plus Main Menu. So here we go.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Program** option.

2. Use the up/down arrow keys select the **Create Program** function and press the return key.

3. Next the screen will display the following message.

This program will let you paint a screen and then automatically create a program from it. Press any key to continue.

4. Press the space bar to continue.

5. The screen will then display the following message.

The entire screen (24 lines and 80 columns) is usable in a TAS program. You may type information anywhere just by pressing the appropriate keystrokes. To move the cursor to a different location use the cursor control keys. Press the 'Esc' key to display the list of options that are available. These options will allow you to add fields, remove fields, add boxes, and remove box3es from the screen. Also, you can place gaphic characters or change the color of different parts of the screen by choosing the color or graphics options and then moving the cursor using the cursor control keys.

Press any key to continue.

---

6. Press the space bar to continue.

7. What you now see is a blank Screen Painter screen. In the lower left hand corner of the screen it will say "Screen/Program Create." In the lower center of the screen are your column and row indicators. These tell you the location of your cursor at all times.

| Screen/Painter Create Col= 1 Row= 1 F1-Help |

Practice using your arrow keys to move the cursor around the screen. Notice that as you move the cursor on the screen, the column and row indicators also change.

8. The TAS-Plus Screen Painter is completely automatic and will write the code for you. You simply place the field titles and the field windows where you want them, answer the editor questions, and TAS-Plus will write the code needed to make the screen work. Of course, you can come back later and rewrite the code using the built-in line editor.

For now you'll just use the arrow keys to move the cursor around and "paint" the screen the way you want it.

9. Move the cursor to column 27, row 2 and type **DEMO - SALES DATABASE**. This is the heading for your screen.

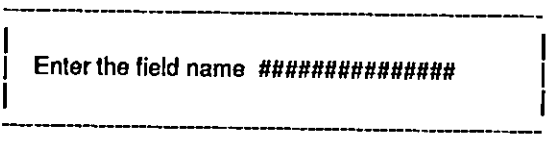
10. Move the cursor to Col 9, Row 6 and type **Order Number:** . This is called a field title.

11. Move the cursor to Col 24, Row 6. Press the **ESC** key to get your Command Window, which will look like this.

```
--- Scrn Choices---------
| A - add field on scrn   |
| R - remove field        |
| S - show field info     |
| J - jump to location    |
| B - box                 |
| P - print scrn          |
| C - color choices       |
| G - graphic chars       |
| X - exit scrn painter   |
-----------------------------
```

12. Press the return key to choose "Add field on scrn".

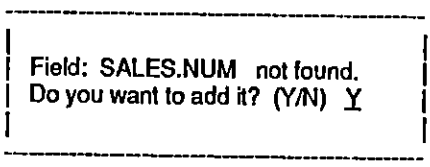
13. A new Command Window will ask for the field name.



Note: Field names can be up to 15 alphanumeric characters.

14. Type in **SALES.NUM** and press the return key.

15. A new Command Window will ask if you want to add this field.



16. Type **Y** for Yes.

17. A new Command Window titled "Enter a New Field" will appear in the middle of the screen. This window lets you specify all the detail about this field. Fill in the answers as below.

```
Enter a New Field

Field Name:          SALES.NUM
Format Name:         SALES
Field Type:          A
  Size:              4
  Dec Chr:
Upper Case (Y/N):    Y
Key (Y/N)            Y

Number of Elements:
  In Array
  - Record

 Array Number
   this Field
Is all of the above correct (Y/N) Y
```

Note: Here are a few notes about the detail shown above. Format Name is the same as Schema is TAS, it allows the same Field Format to be shared by more than one program. Field Type can be (A) Alphanumeric, (N) Numeric, (S) Short Date, (D) Long Date, or (T) Time.

Dec Chr is the number of characters after the decimal and will only be asked if you set the Field Type to numeric. Number of Elements in Array - Record is only asked if the field is not a key field. Just press the Return key to skip over this choice.

18. After you have answered all the questions about the SALES.NUM field, the screen will ask:

Is all of the above correct (Y/N) Y

19. Type **Y** for Yes. If you made a mistake entering your answers, or if you change your mind, you can type **N** for No and start over again at the top. You can do this as many times as necessary to get it the way you want it.

20. After you answer Y, TAS-Plus returns you to the position on the screen where you were before the Command Window

appeared. You should now see a reverse video block that is four characters long. The length of the block equals the number of characters in the field. The < sign at the end of the block lets you know it is a key field.

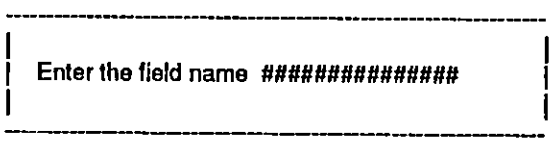
You have now entered your first field title "Order Number:" and your first field window (reverse video block). Later when you run the program, you will input (and view) the appropriate data from your database in this window.

21. Move the cursor to Col 35, Row 6 and type **Name:** .

22. Move the cursor to Col 45, Row 6 and press the **ESC** key to get the Command Window.

23. Press the return key to choose "A - add field on scrn".

24. A new Command Window will ask for the field name.



25. Type **SALES.NAME:** and press the return key.

26. A new Command Window will ask if you want to add this field.

27. Type **Y** (for yes) and you will be shown a new blank entry screen with SALES.NAME as the Field Name.

28. Enter the following information.

```
Enter a New Field

Field Name:              SALES.NAME
Format Name:             SALES
Field Type:              A
   Size:                 25
   Dec Chr:
Upper Case (Y/N):        N
Key (Y/N)                Y

Number of Elements:
  In Array
  - Record

 Array Number
   this Field
```

29. After you have answered all the questions about the SALES.NAME field, the screen will ask:

Is all of the above correct (Y/N) Y

30. Type **Y** for Yes and you will return to the Screen Editor.

31. Move to Col 9, Row 8 and type **Customer Code: .**

32. Move to Col 25, Row 8 and press **Esc** to get the Command Window.

33. A new Command Window will ask if you want to add this field.

34. Type **Y** (for yes) and you will be shown a new blank entry screen with SALES.NAME as the Field Name.

35. Enter the following information.

```
Enter a New Field

Field Name:              SALES.CODE
Format Name:             SALES
Field Type:              A
  Size:                  3
  Dec Chr:
Upper Case (Y/N):        Y
Key (Y/N)                Y

Number of Elements:
  In Array
  - Record

Array Number
  this Field
```

36. Move to Col 35, Row 8 and type **Company: .**

37. Move to Col 45, Row 8 and press **Esc** to get the Command Window.

39. Enter the following information. Be sure to press the return key to skip over "Number of Elements - Record."

```
+-------------------------------------------------------------+
|                                                             |
|      Enter a New Field                                      |
|                                                             |
| Field Name:              SALES.COMP                         |
| Format Name:             SALES                              |
| Field Type:              A                                  |
|   Size:                  25                                 |
|   Dec Chr:                                                  |
| Upper Case (Y/N):        N                                  |
| Key (Y/N)                N                                  |
|                                                             |
| Number of Elements:                                         |
|   In Array                                                  |
|   - Record                                                  |
|                                                             |
| Array Number                                                |
|   this Field                                                |
+-------------------------------------------------------------+
```

40. Move to Col 9, Row 10 and type **Date:** .

41. Move to Col 20, Row 10 and press **Esc** to get the Command Window.

42. Enter the following information. Be sure to press the return key to skip over "Number of Elements - Record."

```
Enter a New Field

Field Name:              SALES.DATE
Format Name:             SALES
Field Type:              S
  Size:                  8
  Dec Chr:
Upper Case (Y/N):        N
Key (Y/N)                N

Number of Elements:
  In Array
  - Record

Array Number
  this Field
```

43. Move to Col 35, Row 10 and type **Address: .**

44. Move to Col 45, Row 10 and press **Esc** to get the Command Window.

45. Enter the following information. Be sure to press the return key to skip "Number of Elements - Record".

```
Enter a New Field

Field Name:              SALES.ADDR
Format Name:             SALES
Field Type:              A
  Size:                  25
  Dec Chr:
Upper Case (Y/N):        N
Key (Y/N)                N

Number of Elements:
  In Array
  - Record

Array Number
  this Field
```

46. Move to Col 9, Row 12 and type **G/L Acct Num:.**

47. Move to Col 25, Row 12 and press **Esc** to get the Command Window.

48. Enter the following information. Be sure to press the return key to skip "Number of Elements - Record".

```
Enter a New Field

Field Name:              SALES.GLNUM
Format Name:             SALES
Field Type:              A
   Size:                 3
   Dec Chr:
Upper Case (Y/N):        Y
Key (Y/N)                N

Number of Elements:
   In Array
   - Record

Array Number
   this Field
```

49. Move to Col 35, Row 12 and type **City: .**

50. Move to Col 45, Row 12 and press **Esc** to get the Command Window.

51. Enter the following information. Be sure to press the return key to skip "Number of Elements - Record".

```
Enter a New Field

Field Name:              SALES.CITY
Format Name:             SALES
Field Type:              A
  Size:                  25
  Dec Chr:
Upper Case (Y/N):        N
Key (Y/N)                N

Number of Elements:
  In Array
  - Record

Array Number
  this Field
```

52. Move to Col 9, Row 14 and type **Amount: .**

53. Move to Col 17, Row 14 and type **$.**

54. With cursor on Col 18, press **Esc** to get the Command Window. Enter the following information. Be sure to press return key to skip "Number of Elements - Record".

```
Enter a New Field

Field Name:              SALES.AMT
Format Name:             SALES
Field Type:              N
   Size:                 9
   Dec Chr:              2
Upper Case (Y/N):        N
Key (Y/N)                N

Number of Elements:
  In Array
  - Record

Array Number
  this Field
```

55. Move to Col 35, Row 14 and type **State: .**

56. Move to Col 45, Row 14 and press **Esc** to get the Command Window.

57. Enter the following information. Be sure to press the return key to skip "Number of Elements - Record".

```
Enter a New Field

Field Name:              SALES.STATE
Format Name:             SALES
Field Type:              A
   Size:                 2
   Dec Chr:
Upper Case (Y/N):        Y
Key (Y/N)                N

Number of Elements:
   In Array
  - Record

 Array Number
   this Field
```

58. Move to Col 57, Row 14 and type **Zip: .**

59. Move to Col 65, Row 14 and press **Esc** to get the Command Window.

60. Enter the following information.

```
Enter a New Field

Field Name:              SALES.ZIP
Format Name:             SALES
Field Type:              A
   Size:                 5
   Dec Chr:
Upper Case (Y/N):        Y
Key (Y/N)                Y

Number of Elements:
  In Array
  - Record

Array Number
  this Field
```

61. Just to dress things, let's paint a box on the screen. Here's how.

62. First move the cursor to Col 6, Row 16, which will be the lower left hand corner of the box. Press the **Esc** key to get the Command Window.

63. Use the arrow keys to move down and choose B - box.

```
--- Scrn Choices---------
| A - add field on scrn  |
| R - remove field       |
| S - show field info    |
| J - jump to location   |
| B - box                |
| P - print scrn         |
| C - color choices      |
| G - graphic chars      |
| X - exit scrn painter  |
---------------------------
```

64. You will see the following message on the screen.

```
-------------------------------------------------------------
| To place a box on the screen first                        |
| press any chr to remove this message                      |
| then using ONLY the arrow keys move                       |
| the cursor to where you want the upper                    |
| right hand corner of the box and press                    |
| ENTER.                                                    |
-------------------------------------------------------------
```

65. Press the space bar to continue.

66. Use the arrow keys only to move the cursor to Col 72, Row 4 and press the return key. A box will appear on the screen.

67. Move to Col 6, Row 3 and press **Esc** to get the Command Window.

68. Press **B** to select "B - box" and press the space bar to continue past the message.

69. Use the arrow keys only to move to Col 72, Row 1 and press the Return key.

70. Your finished screen should now look like this:

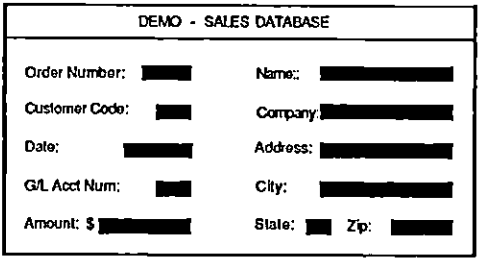


Figure T9 - Sales Input Screen

This completes the screen "painting" phase. All the field specifications that you designated have been automatically entered into the Data Dictionary for you.

TAS-PLUS PROGRAM GENERATOR

The next thing you want to do is save the screen and have TAS-Plus automatically write a program for you. The program will be based on the screen and the information you gave about each field. Again, this is all done automatically for you.

One of the reasons TAS-Plus is so fast to works with is that it combines the Screen Painter, Program Generator and Runtime Compiler all into the **Create Program** function.

Unlike many other "high-level" database programs, TAS-Plus doesn't make you change programs in mid stream or worry about LINKERS, PROCEDURES, LIBRARIES or other cryptic commands.

You have already done all the hard work of producing the database application. In fact, there is only a couple of steps to generating the program.

1. Press **Esc** to get the Command Window.

2. Use your arrow keys to choose "Exit" and press the return key.

3. After a moment, a message will appear on the screen saying:

```
------------------------------------------------------------------
Now please wait while I initialize the file for use
```

4. Next, a new Command Window will ask you if you want to save the screen.

```
------------------------------------------------------------------
| Do you want to save this screen and exit? (Y/N) Y              |
|                                                                |
------------------------------------------------------------------
```

5. Type **Y** for Yes.

6. A new Command Window will ask you for the name of the new screen.

7. Type in **SALES** and press the Return key.

```
------------------------------------------------------------
| Enter the name of the screen SALES                       |
|                                                          |
------------------------------------------------------------
```

NOTE: This name can be up to eight alphanumeric characters long.

8. TAS-Plus will then automatically write the the source code for you. The source code will have a name ending with ".EDT". Our sample program, for instance, will be called SALES.EDT. In a few seconds, after TAS-Plus is finished writing, the following message will appear on the screen.

```
| Enter the name of the screen SALES |
| Creating Program: .EDT             |
```

The next step happens very very fast. If fact, you won't even see it happening. But it is important to know about anyway. It's the step where TAS-Plus compiles the program for you. And by the time you read this, TAS-Plus is already done.

COMPILING YOUR PROGRAM WITH TAS-PLUS

All TAS-Plus programs must be compiled before they can be used. This is true of programs created with the **Create Program** function as well as programs written with the Source Code Editor.

This step is done for you when using the **Create Program** function. The TAS-Plus Runtime Compiler (like the Program Generator) is completely automatic. In fact, it's already been done for you by the time you read this.

If you would like to try out the program, Type **Y** (for Yes). If you would like to quit, type **N** (for No) and you will return to the TAS-Plus Main Menu.

Congratulations, you have just completed your first TAS-Plus database application. The program will be stored on disk along with your database files. You can now run the program any time by choosing the **Run TAS Program** option from the TAS-Plus Main Menu and entering the program name SALES. You can add records to it, delete records from it, and search for records.

VIII. Creating a TAS-Plus Report

TWO WAYS TO CREATE A REPORT

There are two ways to create reports with TAS-Plus. The first way is to use the built-in TAS-Plus Report Writer. This is a quick and easy way to create reports with up to 128 columns complete with totals.

The second way to create a report is to choose the **Edit Program** function which is part of the **Program** option on the TAS-Plus Main Menu. This allows you to go directly to the built-in Source Code Editor/Checker. You would use this method to build sophisticated reports, such as multi-file accounting reports.

> **Note:** TAS-Plus has an all-in-one Report Writer that works much like the TAS-Plus Program Generator. Because of this, it's usually quicker to start all new reports here - even sophisticated ones that will need extensive modification later on.

For our purposes, we'll stay with the **Create Report** function for now. The Source Code Editor will be covered in a later chapter.

The Report Writer which makes up the heart of the **Create Report** function actually writes a TAS-Plus program. This is important, since it means you can save the program and run the report as often as you like. Later on in this manual when you learn how to create menus, you will even be able to make your reports appear on those menu.

The **Create Report** function creates simple columnar reports up to 128 characters wide. You can select a range of records to print and you can define fields to be totaled together. You can also have your reports displayed on the screen, printed on paper or sent to a disk file.

USING THE "CREATE REPORT" OPTION

So let's get started. Again, the best way to learn about TAS-Plus reports is to actually create one. That's what we're going to do in this section. The following section will show you how to use the **Create Report** function to create a report from the information stored in the **CUSTOMER** database file developed in Chapter IV.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Report** option.

2. Use your up/down arrow keys to choose **Create report** and press the return key.

3. The screen will ask you to "Enter the File Name:___."

   Note: At this point you could press **F2** (Function key 2) and TAS-Plus will give you a list of all the files available on the system.

4. Type **CUSTOMER** and press return.

5. The screen will then ask you to "Enter the fields to be displayed? (All/Selected)."

6. Type **S** (for Selected) and press the return key.

7. A window will appear in the middle of the screen with a list of all fields in the file. It will look like this:

```
---- Fields ----------------------
| A - CUSTCODE                    |
| B - CUSTNAME                    |
| C - CUSTCOMP                    |
| D - CUSTADDR                    |
| E - CUSTCITY                    |
| F - CUSTSTATE                   |
| G - CUSTZIP                     |
| H - CUSTAREA                    |
| I - CUSTEXCH                    |
| J - CUSTPHNUM                   |
| K - CUSTYTD                     |
| L - CUSTLASTORD                 |
---------------------------------
```

Fields Chosen Sel Tot

You can use the up/down arrow keys to move from one choice to another. You can also type just the first character of a choice. The fields you choose will then appear on the right hand side of the screen under "Fields Chosen."

> **Note:** The first field on the report must always be a key field. This is the field by which the report will be sorted. CUSTCODE is a key field and was entered as the first field, so the report will be in order of customer code. TAS-Plus will ask different questions depending on the field data type.

8. Type **A** to choose CUSTCODE to be the first field of the report.

9. The screen will then ask "Is this a selection field? (Y/N) Y." Type **Y**.

> **Note:** Defining a field as "selected" means that you will be able to request a range of records at the time the report is printed. For example, you could ask for a report of all CUSTOMERS living in the 98006 Zip Code who had purchased over $500.

10. Type **C** and CUSTCOMP will appear under Fields Chosen.

11. Type **N** to skip the "select" prompt.

12. Type **K** and CUSTYTD will appear under Fields Chosen.

13. Type **Y** for the "select" prompt.

14. Next the screen will ask "Do you want to total on this field? (Y/N) Y." Type **Y**.

> **Note:** Fields that were originally defined as Numeric fields can have their values added together and displayed as a total in a report.

The screen will look like the one shown below.

```
---- Fields ----------------------
| A - CUSTCODE                    |
| B - CUSTNAME                    |
| C - CUSTCOMP                    |
| D - CUSTADDR                    |
| E - CUSTCITY                    |
| F - CUSTSTATE                   |
| G - CUSTZIP                     |
| H - CUSTAREA                    |
| I - CUSTEXCH                    |
| J - CUSTPHNUM                   |
| K - CUSTYTD                     |
| L - CUSTLASTORD                 |
-----------------------------------
```

Fields Chosen	Sel	Tot
CUSTCODE	Y	
CUSTCOMP	N	
CUSTYTD	Y	Y

15. Press **Esc** let TAS-Plus know you are done with field selection.

16. The screen will ask "Is all of the above correct (Y/N) Y ?".

17. If everything looks good, type **Y** at the prompt: "Is all of the above correct? (Y/N) Y." You can start over by typing **N** for No.

18. A window will appear on the screen asking what you want to name the report that is about to be created.

```
| Enter Program Name: CUSTPR                                    |
|                                                               |
|                                                               |
```

19. Type **CUSTPR** and press the return key.

20 Next the screen will ask you for a report title. This is the title which will appear at the top of page when you print the report.

21. Type **Demo Customer Sales** and press the return key.

```
| Enter Program Name: CUSTPR                                    |
| Enter Prg Title: Demo Customer Sales Report                   |
|                                                               |
```

Note: This title will also be inserted in the program listing for identification purposes.

22. Next a message will appear telling you that TAS-Plus is creating the report program.

```
| Enter Program Name: CUSTPR                                    |
| Enter Prg Title: Demo Customer Sales Report                   |
| Please wait while I create the report program.                |
```

The next step is completely automatic. The Report Writer will actually write a standard TAS-Plus program. This is important since it means you can store the report on disk and use it over and over. It also mean that you can edit the report with the Source Code Editor.

23. TAS-Plus will now give you a chance to immediately use your report. A window will appear with the following message.

```
| Do you want to run the program ? (Y/N) Y |
|                                          |
```

24. Type **Y** and press the return key.

25. The screen will now show you the fields you defined as "Select" and let you choose a starting and stopping point for each. The screen will look like this.

```
Selection screen for Demo Customer Sales Report

CUSTCODE
FROM- ###
THRU- ###

CUSTYTD
FROM- ########
THRU- ########
```

26. Press the return key four (4) times to skip the selection process.

27. The screen will now ask "Send this report to the Screen, Printer, or Disk file: S"

28. Type **S** and press the return key twice. This will send the report to the screen.

The finished report will look like this.

```
-------------------------------------------------------------------------
                      Demo Customer Sales Report

                  Tuesday January 1 1980    10:24:07  AM

CUSTCODE CUSTCOMP                          CUSTYTD
---------- ------------------------------ ----------
001        Middle Eastern Import                125.43
002        Pro Musica                          1376.35
003        Little Greenhouse                    150.45
004        La Villa Restaurant                 2484.64
006        Santa Rosa Sports Clinic            2376.76
---------- ------------------------------ ----------
                                               6513.63

Total number of records printed: 5

Press any key to continue _
-------------------------------------------------------------------------
```

29. When you are finished viewing the report, press the return key to continue. This will return you to the TAS-Plus Main Menu.

RUNNING A TAS-PLUS REPORT

After you create a TAS-Plus report and save it, you can use it over and over again in the future. Here's how to run (use) the **CUSTPR** report that you created in the last section.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Report** option.

2. Use your up/down arrow keys to choose **Run report** and press the return key.

3. The screen will ask you to "Enter the program name: " Type **CUSTPR** and press the return key.

Note: If you forget what TAS-Plus reports already exist, here's how to find out. From the TAS-Plus Main Menu, choose the **Run DOS** Command function which is part of the **Run** option. Next, type **DIR *.RUN** and press the return key. This will show you a list of all TAS-Plus programs including the reports. Press **Esc** to return to the TAS-Plus Main Menu.

4. You will now see the "Selection screen." This allows you to select a range of records to be printed. (Remember that when this report was set up, you defined the CUSTCODE field and the CUSTYTD field as select fields). The screen will look like this:

```
---------------------------------------------------------------

Selection screen for Demo Customer Sales Report

CUSTCODE
FROM- ###
THRU- ###

CUSTYTD
FROM- ########
THRU- ########
```

This time, let's say we want to list only customers who purchased over $1,000.

5. Press the return key twice to skip over the CUSTCODE FROM and TO fields.

6. At the CUSTYTD FROM prompt, type **1000** and press **return**. This will cause only records with a value of at least $1,000 in the year-to-date sales field to print on the report.

7. Press the return key to skip over CUSTYTD THRU.

8. The screen will now ask if you want to "Send this report to the Screen, Printer, or Disk file: S ." Type **S** (for Screen) and press the return key.

9. TAS-Plus will now print the report to the screen.

The finished report will look like this.

```
-------------------------------------------------------------------------

                        Demo Customer Sales Report

                    Tuesday January 1 1980   10:24:07  AM

CUSTCODE CUSTCOMP                          CUSTYTD
---------- ------------------------------ ----------
002        Pro Musica                        1376.35
004        La Villa Restaurant               2484.64
006        Santa Rosa Sports Clinic          2376.76
---------- ------------------------------ ----------
                                             6237.75

Total number of records printed: 3

Press any key to continue _
-------------------------------------------------------------------------
```

10. By using different selection criteria , you can effectively use the TAS-Plus Report Writer as a quick and easy "query" system. In this way, you can get information out of your database in many different and useful forms..

11. When you are finished viewing the report, press the return key to continue. This will return you to the TAS-Plus Main Menu.

IX. Getting TAS-Plus files to relate

One of the most powerful features of TAS-Plus is its ability to relate up to 16 database files at once. This ability to "share" information between multiple data files and programs is what makes TAS-Plus a relational database and separates it from the simpler file managers.

> **Note:** TAS-Plus is a "Relational Database" in the modern sense of the word. By that we mean that in today's micro-computer terminology, the term "Relational Database" is accepted as describing a powerful DBMS capable of handling multiple files (as opposed to a file manager which handles only one file at a time). In older mainframe and mini-computer terminology, the term "Relational Database" was used to describe a DBMS that met a specific set of criteria as described by Dr. Codd (one of the first proposers of relational theory for databases).

When we say TAS-Plus can relate multiple files, what we mean is a TAS-Plus program can open up to 16 files at one time and can both write data into those files and read data from those files. This ability is essential if you want to move on to building serious database applications.

There are probably as many ways to use relationals as there are people to use them. We can't begin to cover all the ways in this manual. However, here are a few of the more typical ways you might use relationals.

- Updating multiple files from a single data input screen
- Transferring data from one data file to another
- Developing a new data file based on one or more old data files
- Generating reports based on multiple files

RELATING YOUR SALES AND CUSTOMER FILES

In this section you will learn one of the most basic ways of relating files. In this case, how make your SALES program automatically "lookup" information out of the CUSTOMER file (that you created in chapter 4). This will mean that you can enter sales into the sales database without having to retype each customer's information, since the program will do it for you.

USING THE SOURCE CODE EDITOR

In order to do this, you will need to modify the original SALES program. This will also give you your first hands-on experience with the TAS-Plus Source Code Editor.

1. From the TAS-Plus Main Menu, use your left/right arrow keys to move to the **Program** option.

2. Use the up/down arrow keys to select the **Edit program** function and press the return key.

3. The screen will ask you to "Enter the program Path and Name." Type **SALES** and press the return key.

> **Note:** Since your data files and programs are all in the same subdirectory (or floppy disk), a path name isn't necessary.

4. The screen will now display the SALES program code (from chapter 4) that was created by the Program Generator. The screen will look like this:

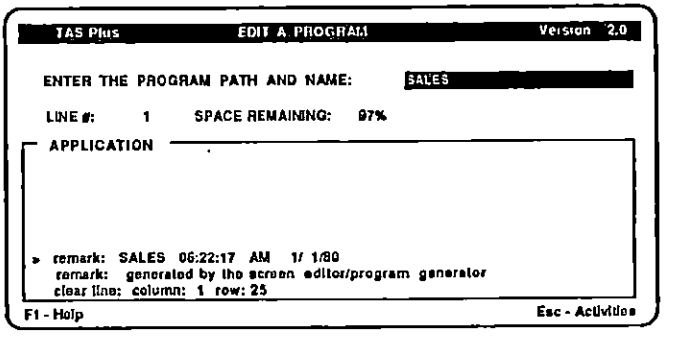


Figure T 10 - Sales program created previously

5. Notice the the small right pointer '>' towards the lower left hand corner of the screen. This indicates the active line. Currently line number one is active.

The first thing you have to do is tell the program to open the CUSTOMER file in addition to the SALES file. Here's what to do.

6. Line 4 shows the statement that opens the SALES data file. Since information is going to read from the CUSTOMER datafile, CUSTOMER will have to be opened also.

4. Use the down arrow key to scroll the screen so that the pointer on the left side of the screen is pointing to the OPEN statement. Notice that the line counter on the top of the screen shows line 4.

5. Press the **Esc** key to get your command window.

6. Use the down arrow key to move down to **I - Insert** on and press the return key.

Two things will happen. One, the first item in the command window will change to **A - add line**. Second, an **INSERT ON** message will appear towards the upper right hand corner of the screen.

> **Note:** Every time you press **I** at the activity window the top selection toggles to the other form. While **A - add line** is active, the words Insert on will appear near the upper right corner of the screen.

7. Press the return key to select **A - add line.**

8. A new Command Bar will appear across the top of the screen and a new command window will appear in the upper left hand corner.

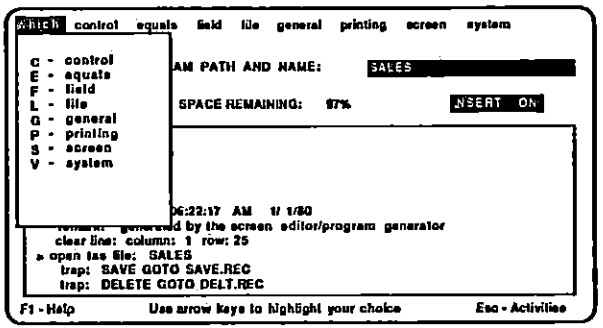


Figure T11 - Command options when adding a record

9. Use your left arrow key to move to the file option.

> **Note:** You could also get to the file option by typing Lwhile in the WHICH command window.

10. Select **O - open tas file** from the **file** command window.

11. The screen will then ask you for the "File name: ." Type **CUSTOMER** and press the return key.

12. The screen will then ask you for the "Company code: ." Press the return key to skip over this question.

13. The screen will then ask for the "Lock Type: ." Press the return key to skip over this question.

14. The screen will then ask for an "on open error goto label: ." Press the return key to skip over this question.

15. The screen will then ask you "Is all of the above correct (Y/N) ? Y ." Type **Y** (for Yes).

Based on the questions that you just answered, the Source Code Editor will automatically insert a new line of code for you.

> **> open tas file: CUSTOMER**

The program will then return you to the command window.

16. Press the Esc key to get back to the right pointer > . This will put you next to line 5.

The next thing you have to do is take out the statement lines that tell the program to enter the customer's name, etc. from the keyboard. These statement lines won't be necessary since you're going to have the program lookup the information in the CUSTOMER file. Here's what to do.

17. Use the down arrow to move to line 11. (Watch the line counter on top). The line looks like this.

> **enter: SALES.NAME**

18. Press the **Esc** key to bring up the command window.

19. Choose **D - delete line.** The line will be automatically deleted for you.

20. Press the down arrow key to move line 12. The line looks like this.

> **enter: SALES.COMP**

21 Press the **Esc** key to bring up the command window.

22. Type **I** to toggle the **A - add line** command to **C - change line.**

23. Press the return key to accept **C - change line.**

What you are going to do now is replace this line with a new one that looks up (relates) the information in the CUSTOMER file. The options presented are Replace, Delete or Edit.

24. Type **R** for **replace command.**

25. Choose **File** from the command window and **F - find record** from the file command window.

26. The edit screen is redisplayed with the new statement in the upper left corner. The statement is shown in reverse video:

find record:

27. Press the return key to choose the default, **E - edit the line.**

28. The screen then asks you "Which type of find." Type an **R** for **Related** find and press return.

> **Note:** The purpose of a related find is to look up information in one file based on information in a second file. Two fields, one in each file, are the reference points that tell TAS-Plus which two records are related and what the common field (or relation) between them is.

29. The screen then asks which "Key field to use." Type **CUSTCODE** and press the return key. This is the field that will be "looked up" in the CUSTOMER file.

30. The screen then asks for the "Related field name:" Type **SALES.CODE** and press the return key. This is the field in the SALES file that will do the "looking up."

Here's how the last two steps will work. When you a actually run the SALES program, you will be asked to enter the customer's code into the SALES.CODE field. At that point, the program will take a look into the CUSTOMER file and try to find a record in which the CUSTCODE field matches the SALES.CODE field. When it finds a matching record, it will copy some of the information from that CUSTOMER record into the SALES record.

31. The screen will then ask "If search error occurs goto line label:" Press the return key to skip over this.

32. Press the return key to verify that the screen is correct.

33. Press **Esc** to exit the command sequence.

As you return to the program listing, you will see that the line for entering the customer name field has been changed to one that finds the customer record with a relational. Line 12 looks like this:

> **find record: MATCH EXACT using field: CUSTCODE using related field SALES.CODE**

Don't worry if the end of the line doesn't show. It's still there.

34. Press the down arrow key once to skip line 13. The right pointer should be next to line 14 which reads.

> **enter: SALES.ADDR**

35. Press **Esc** to bring up the command window.

36. Press **D** to select **D - delete line** (the current line).

37. Press the return key to confirm.

This line, and several others, need to be deleted because they are the commands that cause the program to stop in the customer fields for data entry. Since this program is being edited to read the customer information automatically from the datafile CUSTOMER, you don't want to stop in those fields.

38. Press the down arrow to skip over line 14 **enter: SALES.GLNUM** . The right pointer should be next to line 15 which reads.

> **enter: SALES.CITY**

39. Press **Esc** to bring up the command window.

40. Press **D** to select **D - delete line** (the current line).

41. Press the return key to confirm.

42. Press the down arrow to skipover line 15 **enter: SALES.AMT** . The right pointer should be next to line 16 which reads.

> **enter: SALES.STATE**

43. Press **Esc** to bring up the command window.

44. Press **D** to select **D - delete line** (the current line).

45. Press the return key to confirm.

46. The right pointer should still be next to line 16 which now reads.

> **enter: SALES.ZIP**

47. Press **Esc** to bring up the command window.

48. Press **D** to select **D - delete line** (the current line).

49. Press the return key to confirm.

50. Move the right pointer to line 17 which reads.

> **save record: SALES confirm: Y clear buffer: Y**

51. Press **Esc** to display the command window.

52. Type **I** to toggle to **A - add line** mode.

53. Press the return key to select **A - add line.**

54. Choose **File** from the command window and **C - clear buffer** from the file command window.

55. Type **CUSTOMER** at the filename prompt and press return.

56. Type **B** to clear both the buffer and the record number and press **return.**

57. Press the return key to confirm the entry.

58. The program will add a new statement at line 18 which reads.

> **clear buffer: CUSTOMER RECORD NUMBER AND BUFFER**

Since the information from this record comes partly from the CUSTOMER file and partly from the SALES file, only the left side will be cleared when the SALES record is saved. Clearing the CUSTOMER buffer will clear the data on that side of the screen.

59. Press **Esc** to exit from the edit window.

60. Press **Esc** again to bring up the command window.

61. Choose **S - screen add/chg**.

The command window now displays two choices. The first choice is the only screen name associated with this program (S SALES) and the second is NEW SCREEN. As you add screen and report formats to a program, the new names will be listed here. You can have up to 16 screen and report formats in each program. The last selection on the menu will always be NEW SCREEN.

62. Press the return key to select **1 - S SALES**.

The screen will clear and after a moment the data entry screen for the **SALES** program (created in a previous chapter) will appear.

63. Use the arrow keys to move the cursor to column 45, row 6 (which is the left-most position in the NAME data input block).

64. Press **Esc** and and a window will appear showing several Screen Choices.

65. Choose **S - show field info**.

66. An information window will appear that displays the specifics of the field.

Field Name	SALES.NAME
Array Number	
Type	A
Size	25
Decimal Chrs	
Column	45
Row	6

67. Press any key to remove the definition window.

68. Press **Esc** to bring up the command window.

69. Choose **R - remove field**.

70. The data input block marking the field position will disappear.

71. Press **Esc** for the command window.

72. Choose **A - add a field**.

73. Type **CUSTNAME** and press the return key.

TAS-Plus will look up the definition of CUSTNAME in the Data Dictionary and display a data input block marking the position of the new field.

75. Move the cursor to column 45, row 8.

76. Remove the field **SALES.COMP**.

77. Replace it with the field **CUSTCOMP**.

78. Move the cursor to column 45, row 10.

79. Remove the field **SALES.ADDR**.

80. Replace it with **CUSTADDR**.

81. Move the cursor to column 45, row 12.

82. Remove the field **SALES.CITY**.

83. Replace it with **CUSTCITY**.

84. Move the cursor to column 45, row 14.

85. Remove the field **SALES.STATE**.

86. Replace it with **CUSTSTATE**.

87. Move the cursor to column 65, row 14.

88. Remove the field **SALES.ZIP**.

89. Replace it with **CUSTZIP**.

90. Press **Esc** and choose **X -exit scrn painter**.

91. The screen will ask "Do you want to save this screen ? (Y/N) Y ." Type **Y** for Yes.

92. The right pointer will be next to line 18 and the following lines of the program will be visible on the screen.

```
-----------------------------------------------------------------
   open tas file:  SALES
   open tas file:  CUSTOMER
   trap:  SAVE GOTO SAVE.REC
   trap:  DELETE GOTO DELT.REC
   mount:  SALES type: S
START
   enter:  SALES.NUM
   enter:  SALES.CODE
   find record:  MATCH EXACT using field: CUSTCODE
      using related field:  SALES.CODE
   enter:  SALES.DATE
   enter:  SALES.GLNUM
   enter:  SALES.AMT
SAVE.REC
   save record:  SALES confirm: Y clear buffer: Y
>   clear buffer:  CUSTOMER RECORD NUMBER AND
      BUFFER
   goto:  START
DELT.REC
-----------------------------------------------------------------
```

93. Press **ESC** and choose **V - save program.**

94. The screen will say "Please wait while the program is saved."

95. The screen will then ask "Do you want to compile the program?" Type **Y** for yes.

96. The screen will then ask if you want to "Send the results of the compile to the Screen, Printer, or Disk file: S." Press the return key to accept the default of S.

97. The standard compiler messages will display and then you will be returned to the editor with a blank screen.

98. Press **Esc** to exit the editor and go back to the TAS-Plus Main Menu.

That's all there is to modifying the SALES program so it will relate to the CUSTOMER file. Now let's try the program and see how it works.

USING YOUR RELATED FILES

1. At the TAS-Plus Main Menu, select **Run TAS program.**

2. Type **SALES** as the program name to run and press the return key.

3. The **SALES** data entry screen will display as shown below.

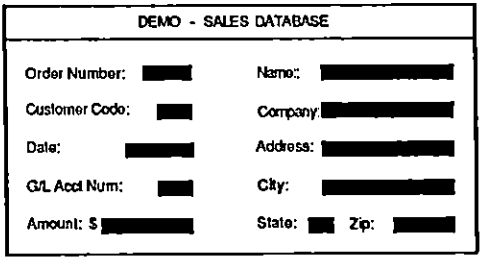


Figure T9 - Sales Input Screen

4. The cursor will start in the first field, which is the Order Number: field.

5. Enter the number **1005** and press the return key.

6. With the cursor in the Customer Code field, type **001** and press the return key.

Watch what happens when you press return. The program automatically checks the Customer Code number and looks for a matching Customer Code number in the CUSTOMER file. When it finds one, it displays the Name, Company, Address, City , State and Zip for you.

This saves you time and keystokes, as well as letting you visually check the entry that was found for the Customer Code to make sure you have entered the correct one.

Finish entering the record as follows.

7. Type in today's date and press the return key.

8. Type **101** as the G/L account number and press the return key.

9. Type **451.25** for the amount and press the return key.

Your screen will look like this.

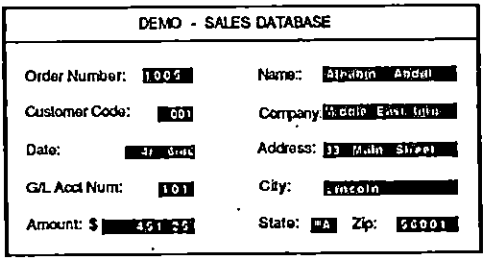


Figure T12 - Sales Input Screen after data has been entered

10. The screen will ask "Save this record?" Type **Y** for yes.

After saving the record, the screen will go blank for a moment, then it will be ready to enter another sales record.

11. For practice, you could enter five or six more sales records at this time. Use the following Customer Codes and TAS-Plus will automatically look them up in the CUSTOMER file for you.

Customer Codes Available:

001
002
003
004
006

12. When you have saved the last Sales record, press **ESC** to exit to the TAS-Plus Main Menu.

X. Creating a TAS-Plus Menu System

The backbone of an application is its menu. Its user-friendliness will be judged partly by users' ease in operating the application from the menus you provide. TAS-Plus has a sophisticated menu command designed to give you maximum flexibility in providing your customers with the kind of pull-down menus popular in today's computer software.

Programming-graphics oriented menus like the ones that TAS-Plus produces used to take days. You'll see in this example that it now takes minutes. In this chapter, you will create a menu to call the programs that you have created in previous chapters. Then you will modify the menu program to take advantage of some other TAS-Plus features introduced later in the chapter.

1. Begin at the TAS-Plus Main Menu. Use the arrow keys to select **Edit program.**

After disk access, the TAS-Plus editor will appear on the screen.

2. Type **SLSMNU** as the program to edit and press the return key.

3. With the pointer on the Remark line, press **Esc** to bring up the activity window.

- Press **I** to toggle to **A - add line** and return to confirm.

4. Choose **Field** from the command window. Choose **Define** from the Field command window.

- Type **CHOICE** as the field name to define and press the return key.
- Type **N** as the field type and press the return key.
- Type **1** as the field size and press the return key.
- Type **0** as the number of decimals and press the return key.
- Press the return key to skip number of array elements.
- Press the return key to confirm the entry.

5. Choose **Field** from the command window. Choose **Define** from the field command window.

- Type **MENULIST** as the field name and press the return key.
- Type **A** as the field type and press the return key.
- Type **27** for the field size and press the return key.
- Type **4** for the array number and press the return key.
- Press the return key to confirm the entry.

When you have defined the two fields, the screen will look like this:

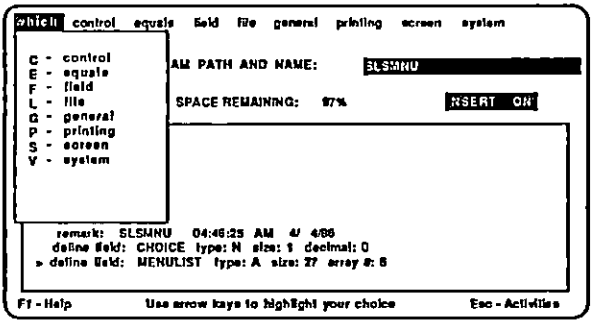

which control equals field file general printing screen system
C - control
E - equals
F - field
L - file
G - general
P - printing
S - screen
V - system
AM PATH AND NAME: SLSMNU
SPACE REMAINING: 97%
INSERT ON
remark: SLSMNU 04:46:25 AM 4/ 4/86
define field: CHOICE type: N size: 1 decimal: 0
» define field: MENULIST type: A size: 27 array #: 6
F1 - Help
Use arrow keys to highlight your choice
Esc - Activities

Figure T13 - SLSMNU program while entering new lines

The menu command uses two fields that must be defined at the beginning of the program. In this sample, the fields are named CHOICE and MENULIST. CHOICE is the field that will store the number of the menu choice selected by the operator. MENULIST is an array field that stores the text that appears on the screen for the menu choices. An array field is different from a regular field in that it is a list of values instead of a single value. The field CHOICE is a single field because it stores a single value -- the position number of whatever was typed in on the menu as the option to execute. MENULIST is a list of values, such as a list you see displayed as menu options.

6. Choose **Equals** from the command window. Press the return key to select the first Equal option.

- Type **MENULIST[1]** for the receiving field and press the return key.

- Type **" C - Customer Program 0102C"** for the expression and press the return key.
- Press the return key to confirm the entry.

This will assign the expression **" C - Customer Program 0102C"** to the first item in the array field MENULIST. The placement of the characters in the expression is crucial because the TAS-Plus menu expects part of it to be at a certain location and derives a lot of information from it.

* the portion of the expression up to (but not including) the first zero provides TAS-Plus the text to display as the menu text (don't forget the space as the first character).

* the next two characters (01) give TAS-Plus the column position in the menu box of the first character of the text (starts in column 1 relative to the upper left corner of the menu box).

* the next two characters (02) give TAS-Plus the row position in the menu box of the first character of the text (row or line 2 - one row down from the top of the box).

> **Note:** In TAS-Plus, when executing a menu you may either move the Bar with the cursor control keys to highlight your choice (and then press the return key) or you may enter the character designated as the choice character (generally the first character in the line of text). This last character is the choice character. Whether the operator enters the choice character or moves the Bar and presses the return key, the value put in the result field (in this case CHOICE) will be the array element number of the line (the number between the square brackets [n]).

* the last character tells TAS-Plus which keyboard character will activate the menu selection

You should decide on the text of your menu choices before you define the first one because they all need to be the same length. **Customer Program** is the longest of the menu choices you will be entering, so the other menu choices should be padded with spaces to equal the same length. Make sure that there are 27 characters and spaces between the quotes.

7. Choose **Equals** from the command window and and press **the return key** to select equals.

- Type **MENULIST[2]** for the receiving field and press the return key.
- Type the expression **" S - Sales Program 0104S"** and press the return key.
- Press the return key to confirm the entry.

8. Choose **Equals** from the command window and and press the return key to select equals.

- Type **MENULIST[3]** for the receiving field and press the return key.
- Type the expression **" R - Customer Report 0106R"** and press the return key.
- Press the return key to confirm the entry.

9. Choose **Equals** from the command window and press the return key to select equals.

- Type **MENULIST[4]** as the receiving field and press the return key.
- Type the expression **" X - Exit to TAS-Plus 0108X"** and press the return key.
- Press the return key to confirm the entry.

Each of the fields in the list (the array) have been set equal to what will display on the menu. The string of characters that you use to describe the menu choice will provide TAS-PLUS the text, location and command character for the choice.

When you are through, the screen will look like this:

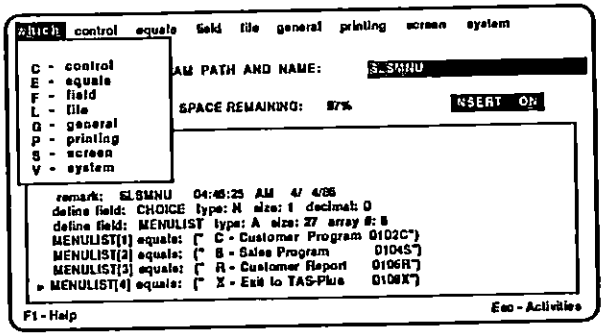

control equate field file general printing screen system
C - control
E - equate
F - field
L - file
G - general
P - printing
S - screen
V - system
AM PATH AND NAME: SLSMNU
SPACE REMAINING: 97%
INSERT ON
remark: SLSMNU 04:45:25 AM 4/ 4/85
define field: CHOICE type: N size: 1 decimal: 0
define field: MENULIST type: A size: 27 array #: 5
MENULIST[1] equate: [" C - Customer Program 0102C"]
MENULIST[2] equate: [" S - Sales Program 0104S"]
MENULIST[3] equate: [" R - Customer Report 0106R"]
MENULIST[4] equate: [" X - Exit to TAS-Plus 0108X"]
F1 - Help
Esc - Activities

Figure T14 - SLSMNU program while entering new lines

10. Choose **Control** from the command window and **Line Label** from the control menu.

- Type **MENU** for the line label and press the return key.
- Press the return key to confirm the entry.

11. Choose **Screen** from the command window and **Clear screen** from the screen command window.

- Press the return key to confirm the entry.

At run time, the clear screen command will clear all text from the screen in preparation for displaying the menu.

12. Choose **Screen** from the command window and and **Menu** from the screen command window.

- Type **30** as the lower left column of the menu and press the return key.
- Type **16** as the lower left row of the menu and press the return key.
- Type **10** as the height of the window and press the return key.
- Type **23** as the menu width and press the return key.
- Press the return key to bypass the multiple column width prompt.
- Type **MENULIST** for the menu lines field and press the return key.
- Type **CHOICE** for the receiving field and press the return key.

- Type 4 for the maximum number of choices and press the return key.
- Type 22 for the maximum choice print width and press the return key.
- Type EXIT as the label to goto and press the return key.
- Press the return key to bypass the help gosub label.
- Type " **Sales Programs** " for the menu title. Be sure to include the quotes and spaces as shown.
- Press the return key to confirm the entry.

When you are done with this step, the screen will look like this:

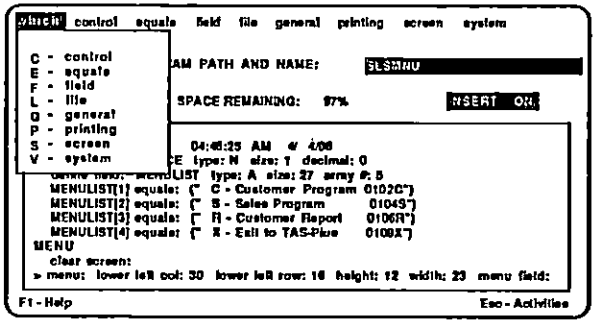

control equate field file general printing screen system
C - control
E - equate
F - field
L - file
G - general
P - printing
S - screen
V - system
AM PATH AND NAME: SLSMNU
SPACE REMAINING: 97%
INSERT ON
04:46:25 AM 4/06
MENULIST[1] equals: (" C - Customer Program 0102C")
MENULIST[2] equals: (" S - Sales Program 0104S")
MENULIST[3] equals: (" R - Customer Report 0106R")
MENULIST[4] equals: (" X - Exit to TAS-Plus 0108X")
MENU
clear screen:
» menu: lower left col: 30 lower left row: 16 height: 12 width: 23 menu field:
F1 - Help
Esc - Activities

Figure T15 - SLSMNU program while entering new lines

13. Choose **Control** from the command window and **On value** from the control command window.

- Type **CHOICE** as the On field and press the return key.
- Type **G** for Goto at the **what to do** prompt and press the return key.
- Type **CUSTOMER;SALES;REPORT;EXIT** for the line label names to goto and press the return key.
- Type **MENU** for the error label to goto and press the return key.
- Press the return key to confirm the entry.

This step lets TAS-PLUS know what part of the program to go to based on the operator's input at the menu. The On value command accepts a fieldname and a list of line labels. When TAS-PLUS executes the line, it branches to the correct line label based on the value of the CHOICE field.

In this case, there are four items on the menu so there will be four possible labels to branch to. If the first menu selection is chosen, TAS-PLUS will branch to the first label listed. If the second menu selection is chosen, TAS-PLUS will branch to the second label listed, etc.

14. Choose **Control** from the command window and **Line label** from the control command window.

- Type **CUSTOMER** for the label name and press the return key.
- Press the return key to confirm the entry.

This sets up a line label to goto from the On value command. Since the label customer is first on the list in the On value command, if the value of CHOICE is 1 (if the operator chooses the first option in the menu box) then the program will branch to this line label and continue processing.

15. Choose **System** from the command window and **Run TAS program** from the system command window.

- Type **"CUSTOMER"** as the name of the program to run and press the return key. Be sure to include the quotes as this differentiates a file actually called CUSTOMER from a field that might be called CUSTOMER.
- Press the return key to confirm the entry is correct.

16. Choose **Control** and **Line label** from the command windows.

- Type **SALES** for the label name and press the return key twice.
- Choose **System** and **Run TAS Program** from the command windows.
- Type **"SALES"** as the program to run and press the return key twice.

17. Choose **Control** and Line label from the command windows.

- Type **REPORT** for the label name and press the return key twice.

- Choose **System** and **Run TAS Program** from the command windows.
- Type **"CUSTPR"** as the program to run and press the return key twice.

18. Choose **Control** and **Line label** from the command windows.

- Type **EXIT** for the label name and press the return key twice.
- Choose **System** and **Run TAS Program** from the command windows.
- Type **"MM"** as the program to run and press the return key twice.

Responding to the name of program to run prompt with am "MM" causes the TAS-Plus Main Menu to be run You could also enter "MENU".

When you are done, the screen will look like this:

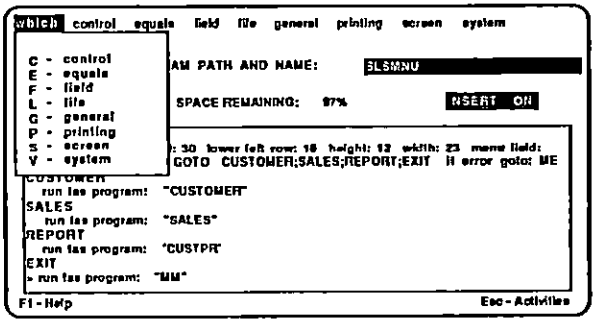


Figure T16 - SLSMNU program while entering new lines

19. Press **Esc** to exit the command window and **Esc** again to bring up the activity window.

20. Choose **Save** program. Press the return key to confirm that you want to compile the program. Follow the screen prompts to press the return key during compiling.

After the new program is compiled, you will the return key to a blank editor screen.

21. Press **Esc** to exit the editor and go back to the main menu.

22. Select **Run a TAS Program** from the main menu. Type in **SLSMNU** for the name of the program to run.

Like magic, your menu appears on the screen. Use the up and down arrows to move the selection bar through the menu. Try each of the selections to see that they call up the programs you've written.

Creating a menu system is easier and faster with TAS-Plus than with any other program or language. Your applications will have a professional polish with a minimum of effort.

Additional Practice

The programs you have listed on your new menu don't know about the menu program you just created. When you quit each of them, they will return you to the TAS-PLUS main menu. You can change each of the three programs (CUSTOMER, SALES and CUSTPR) to return to your new menu instead by adding a few lines to each program.

- First, near the beginning of each program on the menu, enter a trap command of type Q (in the Control section). For the trap action, enter G for goto and EXIT for the label.

- At the end of each program on the menu, enter a line label of EXIT followed by a Run TAS Program command with the program named as 'SLSMNU'. If the program already has an exit label, edit it instead of adding a new one. You can get to the end of the program quickly by pressing the End key (#1 key on the numeric pad).

Save the changes and compile as usual. Then try running the programs and see if they come back to your new menu instead of the TAS-PLUS Main Menu.

XI. Enhancing Your Application

Through Chapter X, you have created a complete TAS-Plus application, including a menu structure. This chapter and the next one will give you additional information and ideas about how to use other TAS-Plus features such as accessing system time and date.

ACCESSING SYSTEM TIME AND DATE

TAS-Plus can read the PC-DOS system time and date and can also set it. In this section, you will modify the menu created in the last chapter to include a selection that displays the date and provides access to change the time and the date.

1. Begin with the TAS-Plus Main Menu on the screen.

2. Use the arrow keys to select **Edit program.**

3. Type **SLSMNU** as the program name at the prompt and press the return key.

After some disk access the TAS-Plus editor will appear on the screen and the program listing for SLSMNU will be displayed.

4. Move the pointer to line 3 of the program.

5. Press **Esc** to bring up the Activity window and press the return key to select **Change line.**

- Press the return key to accept the default action of Edit.
- Press the return key to until you see the Array elements prompt.
- Type 5 and press the return key.
- Press the return key to confirm the entry.

The menu designed in chapter X had four selections on it. This menu will have five selections including the new date selection you are adding.

6. Press **Esc** to exit the edit window.

The program listing will show a value of 5 for the number of array elements in the definition of the field MENULIST.

7. Press **Esc** to display the Activity window.

- Press **I** to toggle to add mode.
- Press the return key to select Add line.

8. Choose **Field** from the command window and **Define** from the field command window.

- Type **CURRENT.TIME** for the name of the time field and press the return key.
- Type **T** for the field type and press the return key.
- Press the return key for the array elements prompt.
- Press the return key to confirm the entry.

TAS-Plus knows that time fields are all 8 characters long, so it isn't necessary to enter a size.

9. Choose Field from the command window and Define from the field command window.

- Type **CURRENT.DATE** for the field name to define and press the return key.
- Type **D** for the field type and press the return key.

TAS-Plus has two types of dates. **D** defines a long date which has 10 characters and includes room for a four-digit year. The second type is S which stands for short date. Short dates are 8 characters long and have a two-digit year.

Just like the time, TAS-Plus knows the date field size for long and short dates.

- Press the return key for number of array elements and the return key to confirm the entry.

10. Choose **General** from the command window and **Date** from the general command window.

- Type **CURRENT.DATE** for the field name and press the return key.
- Type **G** for Get system date and press the return key.
- Press the return key to confirm.

11. Select **Equals** from the command window and press the return key to select the standard equals command from the equals menu.

- Type **MENULIST[5]** for the receiving field and press the return key.
- Type **" D - Date: "+current.date+" 0110D"** for the expression and press the return key.

Previously when you assigned values to MENULIST,they were alphanumeric constants. This time you are using a combination of constants and fields. Alphanumeric fields, date and time fields and numeric fields can all be added together. Since the first part of the expression (" D - Date: ") is alphanumeric, TAS-Plus will convert the rest of the parts of the expression to alphanumeric. Unlike numbers, when you add alphanumeric data together, you get a field that consists of the parts following one another instead of a cumulative total.

For example, if the field CURRENT.DATE were equal to 10/15/86 that expression would equal " D - Date: 10/15/86 0110D".

- Press the return key to confirm the entry.

12. Press **Esc** to exit Add mode.

- Use the arrow keys to scroll until the pointer is on the menu command line.

Be sure the pointer is on the menu command line and not the menu line label.

- Press **Esc** to bring up the Activity window.
- Press **I** to toggle to change mode.
- Press the return key to select change mode.

13. Press the return key to edit the line.

- Press the return key to accept the current values until you see the menu height prompt.
- Type **12** for the menu height and press the return key.
- Press the return key to continue accepting the current values until you see the prompt for the maximum number of menu choices.
- Type **5** and press the return key.

The menu size has changed because a new selection has been added to the bottom. The values of the menu command should reflect the new size.

- Press the return key through the prompts until the edit window for the next command is again displayed on the screen.

14. Press the return key to edit the line.

- Press the return key to accept CHOICE as the On value field.
- Press the return key to accept **G** as what to do.
- Use the arrow keys to move the cursor to the end of the line of labels.
- Type **;DATE** for the label name and press the return key.

Be sure to put the semi-colon before the label name DATE to separate it from the other line label names.

- Press the return key to accept **MENU** as the error label.
- Press the return key to confirm the entry and **Esc** to exit the Edit mode.

15. Use the arrow keys to scroll the screen until the pointer is on line 23.

- Press **Esc** to bring up the activity window.
- Press **I** to toggle to Add mode.
- Press the return key to select Add line.

16. Choose **Control** from the command window and **Line label** from the control window.

- Type **DATE** for the label name and press the return key.
- Press the return key to confirm the entry.

This is the label to goto when the date selection is chosen.

17. Choose **Screen** from the command window and **Clear screen** from the screen command window.

- Press the return key to confirm the entry.

18. Select **Print** from the command window and **Print box** from the print command window.

- Type **20** for the box lower left column location and press the return key.
- Type **14** for the box lower left row location and press the return key.
- Type **6** for the box height and press the return key.
- Type **45** for the box width and press the return key.
- Press the return key to accept **N** as the default for clear the box.
- Press the return key to confirm the entry.

These commands will clear the screen and display a box at run time.

19. Select **Print** from the command window and **Print message** from the print command window.

- Type **"What is the current date? "** for the message to print and press **the return key**.
- Type **25** for the column location of the message and press **the return key**.
- Type **10** for the row location of the message and press **the return key**.
- Press **the return key** to bypass the "Wait for chr entry" prompt.
- Press **the return key** to bypass the "do not enter carriage the return key prompt.
- Press **the return key** to bypass the "force output" prompt.
- Type **CURRENT.DATE** as the entry field and press **the return key**.
- Press **the return key** to confirm the entry.

At run time, TAS-Plus will display a message asking what the current date is at the location specified above. The program will wait until the user has typed in a response and pressed return before continuing with the program. The input the user types in will be assigned to the field CURRENT.DATE.

20. Select **General** from the command window and **Date** from general command window.

- Type **CURRENT.DATE** as the field name and press the return key.
- Type **S** for Set system date and press the return key.
- Press the return key to confirm the entry.

21. Choose **Equals** from the command window. Press the return key to select Equals.

- Type **MENULIST[5]** for the receiving field and press the return key.
- Type **" Date: "+CURRENT.DATE+" 0110D"** for the expression and press the return key.
- Press the return key to confirm the entry.

This will reassign the menu item for date with the corrected date.

22. Select **General** from the command window and **Time** from the general command window.

- Type **CURRENT.TIME** for the time field and press the return key.
- Type **G** to get the system time.
- Press the return key to confirm the entry.

23. Choose **Print** from the command window and **Print message** from the print command window.

- Type **"What is the current time? "** and press the return key.
- Type **25** for the column location and press the return key.
- Type **12** for the row location and press the return key.
- Press the return key for the next three prompts.

- Type **CURRENT.TIME** for the entry field.
- Press the return key to confirm the entry.

24. Select **General** from the command window and **Time** from the general command window.

- Type **CURRENT.TIME** for the time field and press the return key.
- Type **S** for set system time and press the return key.
- Press the return key to confirm the entry.

25. Choose **Control** from the command window and **Goto** from the control command window.

- Type **MENU** as the label to goto and press the return key.
- Press the return key to confirm the entry.

26. Press **Esc** to exit the command sequence.

When you are finished with these changes the program listing will look like the one below. If you are sure that your program listing matches, save the program and compile it as before.

Return to the TAS-Plus Main Menu and run the program **SLSMNU** to see the results.

ADDING A PASSWORD

A password type can be specified in TAS-Plus at the enter command. The difference between a standard Enter command and one that has been designated as a password is that the password type will not display on the screen. Instead, question marks will be displayed on the screen as each character is typed by the operator.

1. Start at the TAS-Plus Main Menu.

2. Choose **Edit program** from the TAS-Plus Main Menu.

3. Type **SLSMNU** for the program name to edit and press the return key.

4. Use the arrow keys to scroll the screen until the pointer is on the last define command.

- Press **Esc** to display the Activity window.
- Press **I** to toggle to Add mode.
- Press the return key to select Add line.

5. Choose **Field** from the command window and **Define** from the field command window.

- Type **PASSWORD** for the field name and press the return key.
- Type **A** for field type and press the return key.
- Type **3** for field size and press the return key.
- Press the return key to bypass array elements prompt.
- Press the return key to confirm the entry.

The field PASSWORD will be used to enter a password on the screen before the menu is displayed.

6. Press **Esc** to exit the edit mode.

7. Scroll the program listing until the pointer is on the Clear screen command after the MENU line label.

- Press **Esc** to bring up the Activity window.
- Press the return key to select Add line.

8. Select **Print** from the Command window and **Print message** from the print command window.

- Type **"What is the password? "** for the message to print and press the return key.
- Type **25** for the column location of the message and press the return key.
- Type **10** for the row location of the message and press the return key.
- Press the return key to bypass the next three prompts.
- Type **PASSWORD** as the entry field and press the return key.
- Press the return key to confirm the entry.

9. Choose **Control** from the command window and **If** from the control command window.

- Type **PASSWORD** for the compared field and press the return key.
- Type **<>** for the type of comparison (this means "not equal to") and press the return key.
- Type **"IBM"** for the expression and press the return key.
- Type **G** for "what to do" and press **the return key.**
- Type **MENU** for the label to goto and press the return key.
- Press the return key to confirm the entry.

The effect of this command is to compare the input the operator entered at the password prompt to the password (IBM). If the comparison is true (in this case not equal to) then the program will loop back to the MENU label. If the comparison fails (the password is correct) then the program will continue processing with the next line.

10. Select **Screen** from the command window and **Clear screen** from the screen command window.

- Press the return key to confirm the entry.

11. Save the program and compile as before. Then try running it to see if it allows access to the programs or prevents access, depending on the password.

Remember, the password is IBM.

XII. Advanced TAS-Plus Topics

USING ARRAYS

Array fields were introduced in Tutorial Chapter X as being fields that are lists instead of single variables. This section will explain arrays further.

Suppose you were counting the students in different types of language classes and only had standard variables to use. The field FRENCH would equal the number of French students, GERMAN would equal the number of German students, etc. With only two or three languages, the task would not be terribly difficult. However, for each language class you add, the more cumbersome the procedure becomes.

```
define: LANGUAGE type: N size: 2
define: FRENCH type: N size: 3
define: GERMAN type: N size: 3
define: SPANISH type: N size: 3
define: ITALIAN type: N size: 3
define: CHINESE type: N size: 3
if: LANGUAGE = (1) then
FRENCH = (FRENCH + 1)
if: LANGUAGE = (2) then
GERMAN = (GERMAN + 1)
if LANGUAGE = (3) then
SPANISH = (SPANISH + 1)
if LANGUAGE = (4) then
ITALIAN = (ITALIAN + 1)
if LANGUAGE = (5) then
CHINESE = (CHINESE + 1)
```

All this repetition can be avoided by using an array field instead of a single field. Keep a list of the languages instead. TAS-Plus can keep up to 255 items in each list (elements of the array). In this example, that would be 255 different language classes. Arrays need to be defined either in the datafile, or in the program as a temporary field.

```
define: LANGUAGE type: N size: 3 array: 5
define: NUMBER type: N size: 3

LANGUAGE[NUMBER] = (LANGUAGE[NUMBER] + 1)
```

If the language number is 1, the field LANGUAGE[1] will be incremented by 1. If the language number is 4, the field LANGUAGE[4] will be incremented by 1.

At the end, you have a list of each of the five language types and how many students are in each one. This saves a tremendous amount of typing and keeps the flow of your programs much simpler.

IF, THEN, ELSE LOGIC

Computers can make decisions quickly and follow complicated paths of logic to do so. The if and else commands are used to tell TAS-Plus which paths to take.

The if command tells TAS-Plus that if a certain condition is true, take the directed action. The conditions that can be true are that one value is equal to another (=), is less than another (<), is greater than another (>), is less than or equal to another (<=), is equal to or greater than another (=>), or is not equal to another (<>).

The action to take can be to goto a line label, gosub a line label, do a procedure (enclosed in braces), reenter the last enter command, return from a subroutine, or execute a single line of the program (then).

Also, TAS-Plus can be told that if the condition is not true that an alternate set of instructions should be followed. This is called the else command.

Ifs and elses can be nested several layers deep to make complicated decisions. An example of the logic of a decision in English would be:

If John's age = Jack's age then:

they have the same birthday

or else:

if John's age < Jack's age then:

John was born after Jack
John is younger than Jack

or else:

John was born before Jack
John is older than Jack

The same exact logic in TAS-Plus syntax would look like this:

```
if: JOHNS.AGE = (JACKS.AGE) DO
{
print message: 'They have the same birthday.'
}
else:
{
if: JOHNS.AGE < (JACKS.AGE) DO
{
print message: 'John was born after Jack.'
print message: 'John is younger than Jack.'
}
else:
{
print message: 'John was born before Jack.'
print message: 'John is older than Jack.'
}
}
```

RULES FOR USING BRACES

The following is a discussion of a type of grouping known as structured programming. Braces ({}) are used in TAS-Plus to group certain sections of program. Some of the TAS-Plus commands need to have the sections of program that affect them grouped. The commands that must be followed by "braced" instructions are: for/next, else and while. The commands if, if carriage return, if duplicate and if rec not found only require "braced" instructions following them if the do option is selected. Braces are not needed for the other option.

The rules for using braces are:

- The open brace ({) must come before the close brace (}).

- For every open brace at the beginning of a procedure, there must be a corresponding close brace at the end of the procedure.

- You may do a gosub from within a procedure and the structure of the procedure will be preserved.

- You may not do a gosub or goto to the middle of a procedure. The gosub or goto label must be a label that is outside a structure (group).

- Procedures may be nested up to 10 levels deep.

- If an up arrow is issued and a close brace is encountered, TAS-Plus will ignore the commands in the procedure until the corresponding open brace is found. This will prevent the commands in a procedure from being executed in reverse order.

- If an up arrow is issued, an open brace is encountered and the previous command is a For/Next, then the loop will be reversed and executed once. This usually has the effect of decrimenting the counter by one. If this action would exceed the beginning of the For/Next counter, then TAS-Plus will ignore the loop and continue looking backward in the program for an Enter command.

Examples of procedures or "braced" instructions:

In the case below, if the value of FLAG is 1 then the procedure following the if command in braces will be executed. After that, TAS-Plus will skip the else procedure and control will pass to the line following (goto: READ.NEXT).

If the value of FLAG is 0, then the procedure following the if will be skipped and the procedure following the else will be executed. After that, the line following (goto: READ.NEXT) will be the next one executed in either case.

```
if: FLAG = (1) DO
{
gosub: TOF
CNTR = (CNTR + 1)
FLAG = (0)
}
else:
{
gosub: PRINT.REC
}
goto: READ.NEXT
```

In the next example, the For/Next command is used. The For/Next command executes a loop a certain number of times. In this case, the loop increments a field called CNTR 10 times. The counter addresses a different array element each time. The first time through the loop, FLD1[1] = FLD2[1] * 1.5. The second time through the loop, FLD1[2] = FLD2[2] * 1.5. When the value of the counter is 10, TAS-Plus stops executing the loop and drops down to the next line of the program (goto: PRINT.TOT)

```
for/next: counter field: CNTR start: 1 stop: 10 step 1
{
FLD1[CNTR] = (FLD2[CNTR] * 1.5)
TOT[CNTR] = TOT[CNTR] + FLD1[CNTR]
}
goto: PRINT.TOT
```

ENTRY OPTIONS

The simplest form of data entry within a program is the enter command used without any options. This is the method used by the TAS-Plus program generator when you are first setting up a program. There are options available in the enter command that make data entry faster and more error free.

Of the 10 entry command options available, only the first one (fieldname) is required. Beyond that you may pick and choose which options you wish to use in your program. The nine other options are described below:

1. Mask string - You may want to limit the operator input to certain characters. The mask string prompt will accept two types of input. You may type in a string of characters surrounded by quotes or type in a field name. For example, ZIP codes are normally stored in alphanumeric type. However, you don't want an operator to be able to type in non-numeric characters. You can type in "0123456789" as acceptable characters for input, or you could assign the value "0123456789" to a field and enter the field name as acceptable input for the mask string.

2. Help Label - TAS-Plus has a programmable help system built-in. The function F1 key is the default TAS-Plus help key. You can specify that if your operator presses the help key while the cursor is positioned in this specific enter field that the program should jump to a subroutine and then return to the originating line of the program The help jump is a gosub and the subroutine should finish with a RET).

Typically, the help subroutine would consist of 1) saving the current screen into a buffer (save screen command), 2) clearing the screen (clear screen command) or opening a window (window command), 3) Printing a context-sensitive help message to the screen (print message command with wait for chr entry option), 4) refresh original screen (redisplay screen command) and 5) return command.

3. Up Arrow Label - You may enter a line label at this option if you would like the program to go to a particular line label instead of to the previous enter command. The action of the up arrow key if this option is left at default is to go backward through the program until an enter command is found.

4. Upcase - If answered with a Y, all characters typed in lower case will be converted to uppercase before they are displayed. This is used often in key fields so that there will not be a problem matching keys that are in upper and lower case.

5. Auto CR - Sometimes data entry is speeded up when the program automatically moves to the next field when the current field is full. This is used usually for fields that have a standard length such as telephone number, customer code and ZIP Code. This option will in effect "press return" for you when the last character of the field has been typed.

6. Password Type - This option will turn any standard entry field into a password type field by printing a question mark every time a character is pressed instead of displaying the character that was typed. This is a kind of data security in that unauthorized personnel cannot obtain the password by seeing what is typed at the password prompt. Using this option does not password protect your program, program lines must be written that actually perform the password protection scheme.

7. Enter at Column Number - A column position of the screen may be entered to tell TAS where to place an enter field. This option is used instead of mounting a screen with the field already on the screen. If column number is left blank, then the field must be on the currently mounted screen.

8. Enter at Row number - A row position of the screen may also be entered. As with Enter at column number, this is used for fields that are not present on the mounted screen. If the row number is left blank, then the field must be on the currently mounted screen.

9. Do not use normal rev video entry - TAS highlights entry fields on mounted screens with a reverse video bar. If you are using the **enter at** option of the enter command, you might want to prevent display of the reverse video bar. Answering Y to this option will prevent the display of a bar as described.

DATE/TIME ARITHMETIC

TAS-Plus knows the clock and calendar backward and forewords.

TAS-Plus can tell you what day of the week a particular date is (equals day command), or what month it is (equals month command). The answer to either day or month could be a number or a name depending on what type of field the receiving field is. If the receiving field is alphanumeric, TAS-Plus will set it equal to the name. If the receiving field is numeric, TAS-Plus will set it equal to the number.

define: DAY type: N size: 1
define: MONTH: type: A size: 3
DAY equals day number/name: DATE
MONTH equals month number/name: DATE

In this example, if the date were 3/17/1986, DAY would be 0 and MONTH would be MAR.

Dates can also be added and subtracted, and new dates determined. For instance if you add 30 to a date field that has a value of '4/1/1986', you get a date field with a value of '5/1/1986'.

DATE2 = (DATE1 + 30)

You can subtract 1 from a date field with a value of '3/1/1986' and get a resulting date field with a value of '2/28/1986'. Of course, if it had been a leap year, you would have gotten '2/29/1986' instead.

DATE2 = (DATE1 - 1)

If you subtract a date field with a value of '9/15/1986' from a date field with a value of '9/30/1986', you get the number 15.

INTERVAL = (DATE1 - DATE2)

Time works the same way. You can add time in seconds to another time and get the result. 13:00:00 plus 300 seconds (5 minutes) is 13:05:00.

TIME2 = (TIME1 + 300)
or
TIME2 = (TIME1 + (5 * 60))

Subtracting time also works.

SORTING WITH INDEXES

TAS-Plus maintains a key file or index file that contains all of the indexes for a datafile. This index file is opened for you when you open a datafile and all keys are automatically updated whenever you add, change or delete a record. The index file does not count as an open file in considering the maximum number of files open. The maximum refers only to master data files.

To access a file in key order from the keyboard, place your cursor in the field that you want the data sorted by and press the Begin Rec key. The field must be a key field (generally designated by being preceeded by a > or followed by a <. The first record alphabetically in that file will appear on the screen. If you move to another key field and press the Begin Rec key, the first record by order of the new field will be displayed.

The same sort of file access can be programmed with the Find Record command. In a customer list for example, you might have one type of printout that lists your customers in alphabetical order and another report that lists them by order of year-to-date purchases. Both name and YTD purchases would have to be key fields.

For the first report, you would issue a find record command with option B for beginning record and name as the field to use. After printing that record, you would issue a Find Record with the option N for next record. Since you have already told TAS-Plus that the order you want it by is name in the beginning record line, it will give you the next record in the file alphabetical.

In the second report, you would follow the same steps, but you would use the field YTD instead of name. The report that would be produced then would be by order of YTD purchases starting with the lowest going to the highest. You can also print the report in the reverse order, with the highest purchases at he top of the list. Instead of starting with the begin option followed by next record commands, start with an end option followed by previous record commands.

The third method of programmed access by keys uses the overlay key capability. Sometimes you might want a file ordered by more than one field. For instance, in a large name and address database, you might have several repeated last names. In that case you would want to sort by last name and first name. In the data dictionary, set up the last name and first name the way you normally would.

After saving the record layout, bring it back up so that you can see the offset calculations. Add a field to the dictionary with a length equal to the lengths of the fields that are to be overlayed added together. Be sure to use the Csize or calculated size in figuring the length. Type Y in the overlay column. Usually TAS-Plus skips over the offset column and calculates the offset itself. If you answer Y in the overlay column, TAS-Plus will stop in the offset column and wait for your input. The offset of an overlay is equal to the offset of the first field in the overlay. Answer Y in the Key column so that an index will be set up on this field.

Rules for using overlays:

1. The fields to be overlayed must be contiguous in the datafile.

2. The length of the overlay is equal to the combined lengths of the fields to be overlayed. The size used for determining the length is the Csize column or calculated size.

3. The offset of the overlay is equal to the offset of the first field in the overlay.

4. The overlay field must be defined as a key field if it is to be used for data access order.

MULTIPLE COMPANIES OPTION

TAS-Plus has the capability of using separate datafiles for different companies. This provides you the advantage of maintaining independent databases for unrelated companies rather than combining them in a single file.

TAS-Plus distinguishes between master and key files for different companies by looking for a unique two-character extension at the end of each filename. The standard convention for TAS-Plus master and key files is to end a master file with a .M and a key file with a .K. If you choose a company code of 01, the master file will instead end with a .M01 and the key file with a .K01.

1. The files need to be set up for this method before it is invoked. To make copies of the datafiles with the appropriate extensions, use the DOS copy command. In this example we use a company code of 01. The company code may be any combination of letters and numbers.

```
COPY *.? *.?01
```

2. This copies all .M and .K files with new extensions of .M01 and .K01. It is not necessary to have more than one copy of the data dictionary (TASDICT.M and TASDICT.K), so after you make the copies you can erase TASDICT.M01 and TASDICT.K01.

3. The previous steps are one-time setup procedures. To invoke multiple company mode, each time you load TAS-Plus you must specify which company code will be used. To do this select the utility Select Company Code from the TAS-Plus Main Menu. At the prompt, enter the company code you wish to work with, in our example it would be 01. When this is done, TAS-Plus will

automatically access only datafiles with the specified company code extension.

4. If you wish, you may specify that one or more datafiles be opened with the company code extension in your program. Please see the company code command in the reference section for further explanation of this.

TAS-PLUS

REFERENCE GUIDE

I. Conventions in TAS-Plus Programming and Documentation

ADDRESSING ARRAYS

In the command explanations, the term fieldname[ss] is sometimes used. This means that either a fieldname or a fieldname with an array subscript [ss] is allowed. If there is no [ss] following the word fieldname, then array fields are not a valid response. Please see Tutorial Chapter X for further discussion of arrays.

In the Menu command, the [ss] is omitted because the field must be an array field.

QUOTATION MARKS

In all cases of TAS-Plus programming, single quotes (') are interchangeable with double quotes ("). In the documentation we always use double quotes for consistency, but single quotes may be substituted.

Data to be converted from a comma delimited file which has quotes around fields with embedded commas must use double quotes instead of single quotes. This provision allows apostrophes to be used in names (such as O'Connor).

RESPONDING TO PROMPTS

Two types of input are generally allowed in response to prompts in the editor. The two types are a constant or a field. A constant is a character or group of characters that do not change and are interpreted by TAS-Plus to be the same throughout the program.

ALPHANUMERIC CONSTANTS

Alphanumeric constants are surrounded by quotes (except as noted below) to let TAS-Plus know that it is a constant as opposed to a field name. Alphanumeric constants may be composed of any arrangement of number, letters, punctuation and symbols with the exception of a double quote (") or single quote ('). An example of a constant is "New York City, New York".

NUMERIC CONSTANTS

Numeric constants do not require quotes. Their composition is limited to the digits zero through nine, a decimal point (.) and negative sign (-). Examples of numeric constants are 100, 0.005, -30, -0.1, and 20.5.

> **Note:** Numeric constants must have at least one numeric digit to the left of the decimal point.

FIELDS

Fields are items whose values change or vary depending on what is happening at a particular point in the program processing. A field is given a name of 15 characters or less, for identification purposes, and is defined either in the data dictionary or in the program itself using the Define Field command.

A fieldname may consist of any combination of characters except a backslash (\), spaces or mathematical operators (+,-,/,*). Fieldnames should be chosen for their relation to what they represent. For example, a field that is going to be a column value in a command might be called COL.VALUE or COLUMN_NUM.

In cases where you have the option of responding to a prompt with a field, it will be noted in the documentation that a constant or fieldname is acceptable input.

If the documentation does not refer to fieldname as an acceptable response, as is often the case with filenames, then a constant is the only entry that can be made. Datafile names must always be

constants. In those cases, quotes are not required around the alphanumeric constant because a constant is the only valid response.

If a numeric field will hold a negative value, it must be defined in the program or in the data dictionary as being an odd number of characters in length. This applies both to fields defined in the data dictionary and fields defined in programs. For example, a field defined with a size of 9 can hold the negative value -100 whereas a field defined as 10 characters long cannot.

Numeric fields that have decimal places must have the number of decimals defined as an even number (2, 4, 6 or 8). Also, you must define the field as being at least one character longer than the number of decimal characters. Since the decimal itself is not included when determining the size of a numeric field, a field defined as having a length of three with two decimal characters would appear as 0.00.

FILENAMES AND LINE LABELS

Datafile names and program names are limited to eight characters because of the DOS limitation. All printable characters may be used in the name except spaces, asterisks (*), backslashes (\), question marks (?) and periods (.). Master files (datafiles) have extension of .M and key files (index files) end with a .K. Programs have an extension of .EDT (in source form) and .RUN (in executable form).

Line labels may be up to 10 characters long and may contain any printable character except spaces.

II. Schematic of TAS-Plus Language

TAS-Plus Command Summary

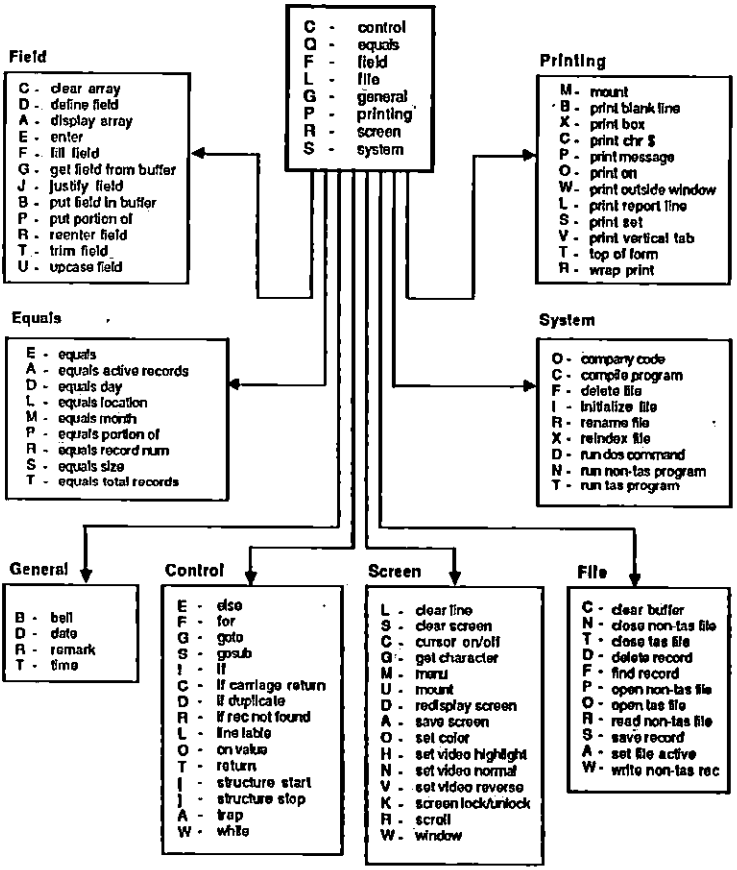


FIGURE R1 Summary of Commands

III. Main Menu Explained

RUN TAS PROGRAM

This selection executes a TAS-Plus program. It will prompt you for the location and filename. To run a program on the current directory of the current disk drive, type in the filename and press return (for example, MYPROG). Other options are 1) to precede the filename with another disk drive designator (for instance A:MYPROG) and 2) to precede the program name with a directory path (for instance, \OTHERDIR\MYPROG). A drive designator and directory path may both precede a program name (for example, A:\OTHERDIR\MYPROG).

TAS-Plus programs must be compiled before they will operate. Also, the TAS-Plus language (TASPLUS.EXE) and the locator files (TASFILE.M and TASFILE.K) must be present on the current disk drive and directory.

RUN NON-TAS PROGRAM

TAS-Plus can run other programs such as a word processor or assembly language routine. The program run by TAS-Plus must have an extension of .COM or .EXE. It will prompt you for the location and filename. When you type the filename, you must include the file extension, either .COM or .EXE. To run a program on the current directory of the current disk drive, type in the filename and press return (for example, MYPROG.COM). Other options are 1) to precede the filename with another disk drive designator (for instance A:MYPROG.COM) and 2) to precede the program name with a directory path (for instance, \OTHERDIR\MYPROG.COM). A drive designator and directory path may both precede a program name (for example, A:\OTHERDIR\MYPROG.COM).

RUN DOS COMMAND

DOS commands can be executed through TAS-Plus. The DOS file COMMAND.COM must be present on the current drive and directory to run a DOS command. Type in the command, including wildcards, the same way you would through MS-DOS. The screen will clear and you will see whatever messages normally appear from DOS. When the action is finished, press any key to return to the TAS-Plus Main Menu. If COMMAND.COM is not present, the screen will clear, but nothing will display. If this happens, press any key to return to the TAS-Plus Main Menu.

CREATE PROGRAM

This selection will let you create a simple program very quickly. You paint a screen and save it. TAS-Plus will write the program for the screen, compile it and given you the chance to run the program at that time.

Programs created with this selection may be enhanced later by running the Edit program selection. For more information on the screen painter, please see Reference Chapter VI.

EDIT PROGRAM

This selection runs the TAS-Plus Editor to begin a new program or change an existing one. It will prompt you for the location and filename. To edit a program on the current directory of the current disk drive, type in the filename and press return (for example, MYPROG). Other options are 1) to precede the filename with another disk drive designator (for instance A:MYPROG) and 2) to precede the program name with a directory path (for instance, \OTHERDIR\MYPROG). A drive designator and directory path may both precede a program name (for example, A:\OTHERDIR\MYPROG). No extension is necessary because only files with a .EDT extension may be loaded in the editor.

A new program may be created either with the Create a program selection or with this one. Create a program is an automatic procedure of creating a screen, program generation and

compiling. If you type a new program name at the prompt for Edit a program, the editor will create a blank program file for you to develop.

COMPILE PROGRAM

TAS-Plus programs written with the TAS-Plus Editor or a word processor are compiled with this selection before they can be executed. It will prompt you for the location and program name. To compile a program on the current directory of the current disk drive, type in the name and press return (for example, MYPROG). Other options are 1) to precede the name with another disk drive designator (for instance A:MYPROG) and 2) to precede the program name with a directory path (for instance, \OTHERDIR\MYPROG). A drive designator and directory path may both precede a program name (for example, A:\OTHERDIR\MYPROG).

The name of the program to compile must end with a .EDT or .SRC. .SRC files are the kind of source code files used in previous versions of TAS.

You may enter "S", "P" or "D" to have your compiler messages printed on the screen, printer or disk file. TAS-Plus will display a compiler sign-on message followed by "Pass 1" and "Pass 2". If there are any errors are located in your program during compiling, messages will be sent to the screen, printer or disk file. An error message includes the line number in the program, the error type and the incorrect item enclosed in angle brackets (<>). The error messages are listed in Appendix X. The most typical error is a fieldname that was spelled wrong so the compiler cannot recognize it.

A message "No Errors" indicates that the compile was successful and you now have a .RUN file that can be executed by the TAS-Plus language.

> **Note:** Before a TAS-Plus program can be run, it must be successfully compiled. If an attempt is made to run a program before it is successfully compiled you will get the error message "Run program does not exist" or "Bad run program".

CREATE DATABASE

A new database may be defined with this selection. After the database is defined, it will be intialized automatically and then you can add data if you wish.

MAINTAIN DATABASE

This option is used to enter, change or delete current records in a database that has been previously created. This is a very useful program that will permit you to enter data into a file without having to create a program for it. After data is entered into a file, you may change or delete any record quickly and easily or add new records.

You may choose the fields to be used or specify to use all fields. If a database has array fields in it, the program will only display the first element in the array unless you choose the specific elements to be used. Fields that are longer than 58 characters, the space allowed on the screen, will automatically wrap around to the next line so that you may see and edit the entire field. Fields that are keys are designated with a right angle bracket (>) in front of the field name (for example, >FIELD.NAME).

Only 15 fields appear on the screen at one time. If you have long fields that have wrapped to the next line, you will see fewer than 15 fields displayed at a time. You can use the arrow keys to scroll up or down to fields that are out of sight. The return key has the same effect as the down arrow key.

When you reach the last field, you will be asked if you want to save the record. You can save the record without moving to the last field by pressing the F10 key.

A record may be deleted from the datafile by finding it (so that it displays on the screen) and then pressing the F4 key. You will be prompted to confirm that the record is to be deleted.

Records that do not have key fields may be entered using this option but may not be retrieved. You may view them in record order by using the Browse Database option or by writing a program.

Any fields left blank, including those that you did not choose for display, will be filled with blanks or zeros. This also applies to key fields.

BROWSE DATABASE

The Browse selection displays records in a database in table format. This means one record per row and a maximum of 10 records on the screen at any time. You may select which fields are to be displayed or allow the program to display all of the fields in a record. If the database has array fields in it the program will display only the first element of the array, unless you choose the specific elements to be displayed.

If you allow the program to display all fields, the records will display in record number order. Record number order is the physical order in the file, which may be different than the order in which the records were entered.

If you select certain fields to be displayed, you can select a key field by which to order the records. You can also select a beginning value to start with. You may access records not visible on the screen by using the down arrow and up arrow keys to scroll through the records. If the record length is longer than the screen is wide, you may access fields that are off the screen by using CTRL-right arrow and CTRL-left arrow to scroll horizontally through the fields.

You may change data in a file by typing new information and pressing return. Using the right and left arrow keys will move the cursor inside a field for editing. When you move to a new record by pressing the up or down arrow keys, if you have changed any data in the record TAS-Plus will ask you if you want to save the record. If you type N at the save record prompt, you will notice that the data does not immediately revert to its original contents. When you move the cursor back to that record, or when the record is scrolled off the screen and back on, the original data will be intact.

You may not add new records or delete existing records in Browse database mode. Use Maintain database to add or delete records.

CREATE REPORT

This selection will prompt you for the filename and specific fields to be included in the report and then create the program for you. You then have the option of running the report. The report program that is created is a standard TAS-Plus program that may be edited with the Edit a program selection to customize it and is stored on the default drive and directory.

RUN REPORT

You may run a report program that was created with the TAS-Plus Editor or with the Create report with this selection. It will prompt you for the location and filename. To run a report on the current directory of the current disk drive, type in the filename and press return (for example, MYREPT). Other options are 1) to precede the filename with another disk drive designator (for instance A:MYREPT) and 2) to precede the name with a directory path (for instance, \OTHERDIR\MYREPT). A drive designator and directory path may both precede a program name (for example, A:\OTHERDIR\MYREPT).

SET DATE

The DOS system date may be set with this selection. You will be prompted for the current date.

INITIALIZE DATABASE

New data files must be initialized before data can be stored in them. This selection will create the new master and key files. Since initializing sets the number of records in a file to zero, you can also use this option to effectively erase a data file. Datafiles created with Create database, Create program and Edit program are initialized automatically and don't need to be intialized again using this option.

REINDEX DATAFILE

This selection builds a new index or key file from the existing data. It may be used if the original index file is missing or damaged. It is also used to create a new index file if you have changed the key structure of a data file.

First, initialize the datafile, keeping a careful note of the number of records in the file as displayed on the screen. Then reindex the datafile and enter the same number of records when prompted.

If your data file has become damaged (evidenced by a "Master File Read Error" or "Key File Read Error"), try reindexing with one or two records less than the actual number of records. Reindexing works in reverse order, starting with the end of the file working toward the beginning.

RESTRUCTURE DATAFILE

Very often when you are creating a database, you will not know exactly what fields are to be used or just what size they might be. TAS-Plus allows you to design your database with flexibility in mind. If you need to add new fields, change the size of current

fields or even remove fields, it can be done quickly and easily using this option.

The first step is to change the current record layout. Use the Maintain data dictionary option to do this. Maintain data dictionary is explained in Reference Chapter IV.

After completing the changes to the record layout in the data dictionary, select this option, Restructure datafile. TAS-Plus will ask you for the filename. Type the datafile name and press return. TAS-Plus will automatically read your old data, make allowances for the new structure and rewrite the data accordingly. You will return to the Main Menu upon completion.

If you made any changes to the keys in the datafile, you should then use the Initialize datafile and Reindex datafile options to set up the new key structure. Both of the options are described in this section. When you Intialize the datafile, please make a note of the number of records in the file. You will need this number for the Reindex datafile routine.

Any program you have written with the Create program, Create report or Edit program selections should be recompiled immediately. You may use Maintain database or Browse database without any further changes.

MAINTAIN DATA DICTIONARY

The TAS-Plus data dictionary stores the record layouts for all of the files in a database. When you create new fields throughout the TAS-Plus system, the information on the structure of those fields is entered for you in this file.

You may change the structure by adding fields, deleting fields, altering fields and changing keys. This selection will give you access to the data dictionary to make those modifications.

MAINTAIN LOCATOR FILE

The TAS-Plus locator file stores information that tells TAS-Plus where to find datafiles. Entries are made automatically in the locator file when you create new fields. The default location stored is the current disk drive and directory. Use this selection to specify a different disk drive and/or directory path for TAS-Plus to locate your datafiles.

When you run this selection, the screen that is displayed is a table in the same form as Browse database. There are three columns, one for each locator field, and a row for each datafile in the data dictionary. The locator fields are: BTLFNAME (File name), BTLSCHEMA (layout name) and BTLMLOC (file location).

All of the cursor control commands work for this selection as described in the Browse database section above. To change the location of a datafile from the default drive to drive B, first type the name of the datafile at the filename prompt for this choice. Then select the row that lists the filename for which you want to change the location. Move the cursor to the BTLMLOC column, type B: and press return. You will see a message asking if you want to save the record. Type Y and press return.

Another example: To change the location of a datafile to a directory on drive C called MYDIR, follow the same procedure, but substitute C:\MYDIR\ for the B: in the example.

The benefit in having a locator file is that it is easy to move datafiles to different disk drives and directories without having to edit and recompile the programs that use that datafile.

Press Esc to exit the locator file and return to the main menu.

CONVERT TAS TO DBASE

This selection will allow you to convert TAS-Plus datafiles into dBase datafiles. You will be prompted for the name of the TAS-Plus file to convert. After checking to make sure the TAS-Plus file exists, you will be prompted for the disk drive, directory path and filename for the dBase file. A file extension is not necessary

because TAS-Plus will assign a .DBF extension. The file will transfer automatically and you will be returned to the TAS-Plus Main Menu.

Alphanumeric, numeric and date fields will transfer to their respective dBase counterparts. Time fields, however, will convert to the dBase character type because dBase doesn't support time data types.

CONVERT DBASE TO TAS

This choice will let you convert dBase files to TAS-Plus format datafiles. You will be promted for the TAS-Plus filename. After checking to make sure the file doesn't already exist in the data dictionary, you will be prompted to enter the dBase disk drive, directory path and filename. File extensions are not necessary as TAS-Plus knows what to look for.

TAS-Plus will automatically update the data dictionary with the dBase field names and types. Character fields are converted to alphanumeric, logical field types are converted to one character alphanumeric, dates are converted to short date types and memo fields are converted to alphanumeric (although the actual data in the memo fields is not converted).

EXIT TO DOS

This selection returns you to the operating system.

IV. Maintain Data Dictionary

Maintain data dictionary works in a way similar to Browse datafile. You will be prompted for the name of the file that holds the records you want to inspect. You may either type in a filename or press F2 to get a list of datafiles to choose from. The fields will be displayed as rows in the order in which they are stored.

A maximum of 10 rows of field descriptions appear at one time. If the record has more than 10 fields you can use the down arrow to scroll down through the data. The up arrow works in the opposite direction. The left and right arrows will move the cursor through the field for editing. To edit a field, type over the existing data and press return.

A field can be deleted by placing the cursor block on the field type designator and pressing the F4 function key. TAS-Plus will confirm that you want to delete the field and, if the answer is Y, the field will be deleted. If this is the first field in the layout, the program will ask if you want to delete the entire layout.

You can add a new field to the record by pressing the F2 key. TAS-Plus will find the last field of the layout and place the cursor block immediately after it. You can then enter information for an entirely new field. If this is an overlay field, it will be sorted into the proper order when the record is saved. The next time you display the layout information, the overlay field will immediately follow the field with the same offset as the overlay.

Example of record layout for the file CUSTOMER entered in the tutorial:

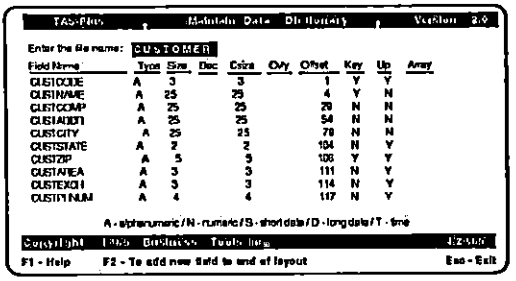

Enter the file name: CUSTOMER
Field Name Type Size Dec Csize Occ Offset Key Up Array
CUSTCODE A 3 3 1 Y Y
CUSTNAME A 25 25 4 Y N
CUSTCOMP A 25 25 29 N N
CUSTADDR A 25 25 54 N N
CUSTCITY A 25 25 79 N N
CUSTSTATE A 2 2 104 N Y
CUSTZIP A 5 5 106 Y Y
CUSTAREA A 3 3 111 N Y
CUSTEXCH A 3 3 114 N Y
CUSTTYNUM A 4 4 117 N Y
A - alphanumeric / N - numeric / S - short date / D - long date / T - time
F1 - Help
F2 - To add new field to end of layout
Esc - Exit

Figure 72 - Maintain Customer Dictionary Record Layout

The following is an explanation of the fields displayed:

Name - The name that you assigned to the field.

Type - The field type may be A - Alphanumeric, N - Numeric, S - Short date (8 character date), D - Long date (10 character date) or T - Time (8 character time).

Size - The defined length of the field. In numeric fields, the size is the number of characters including decimal places, but not including the decimal point. Numeric fields must be defined with an odd size (3, 5, 7, etc.) to hold a negative number.

Csize - The calculated size of the field which is the actual number of characters used to store the field in the file. In the case of alphanumeric fields, the size and csize are the same. The csize of numeric fields is approximately half of the defined size, dates have a csize of 4 characters and time fields have a csize of 3 characters. The total of all the csizes gives you the record length.

By using a special process for storing information called BCD (binary coded decimal) TAS-Plus is able to pack much more information into the same space as most other database managers and makes much more efficient use of your data storage.

Dec - The number of decimal characters in a numeric field. This must be an even number (2, 4, 6 or 8)

Ovly - Whether or not (Y/N) this is an overlay field. Normally the offset value is calculated for you. If this is a Y then you will be able to enter the offset value. An overlay field is used if you want to combine two or more contiguous (next to each other) fields into another field. This generally applies when you want to make a key field from two or more non-similar fields. For example, you could make an overlay key field from customer code and date. You can then order the data retrieval first by customer code and then date within customer code.

To use an overlay field, first set up your record with the fields to be overlayed next to each other. Save the record layout so that the offsets will be calculated. Then use this choice to add the overlay field. You might call the field CUST.CD.KEY and specify the offset value as the same offset you see displayed for the first field, CUSTCODE. The length of the overlay field is the combined Csize (not size) of the fields used in the overlay.

TAS-Plus will allow you to use the first field of the overlay as a key, even though is is not designated as one.

Offset - The position of the field in the record. An offset value of 1 is the first position within the record. This column is skipped unless the field is an overlay field.

Key - Whether or not (Y/N) this field is designated as a key field. Up to 16 different fields can be designated as key fields in TAS-Plus.

Up - If this is an alphanumeric field you may specify whether or not (Y/N) all the characters should be converted to upper case or left as mixed cases.

Array # - The number of array elements for this field. Any non-key field may have from 0 to 255 array elements.

Note: It is important that you do one of two things if you change a record layout. If the file has data in it that you don't want to loose, use the Restructure datafile option from the Main Menu. If it is a new file, or one that has data you don't want to keep, then use the Initialize datafile option from the Main Menu.

V. Source Code Editor

TAS-Plus has a completely menu-driven source code editor. The editor can be used to change source code generated automatically by TAS-Plus through the Create program or Create report selections. It can also be used to create applications from scratch. The editor creates source code files that end with a .EDT extension. The source code can then be compiled into a .RUN file and run with TAS-Plus.

To load the source code editor, choose Edit program from the TAS-Plus Main Menu. When prompted for the program name, either type the name of an existing program to edit or a new name. For example, MYPROG. It is not necessary to give the .EDT extension with the filename. If you enter the name of an existing program, the program listing will be displayed on the screen. If you type a new name, TAS-Plus will create a program with that name and give you the first line of the program, a remark line.

You may precede the program name with a drive designator and directory path if the program is not on the current drive. For example, C:\TASDIR\MYPROG.

The editor screen has several parts. At the top you will see the program name, the active line number and the amount of memory space remaining for the program. The middle portion is the application window where the program listing is displayed. On the left side of the application window a pointer indicates the active line. At the bottom of the screen are reminders that you press F1 for help and Esc to exit the current mode. When the editor comes up, it is in display mode. Pressing Esc brings up the activity window shown below. It lists the options available.

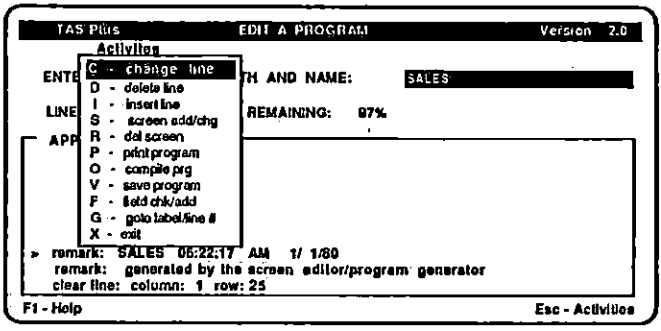


Figure R 3- Editor Activity Window

CHANGE/ADD LINE

The first option in the activity window toggles between Change line and Add line. To switch from one to the other, press I for insert. While in change line mode, selecting this option displays the active line in an edit box so that you can review and make changes if necessary. You may change the line completely, part of it or even delete it. You can change the active line by using the arrow keys to scroll through the program. The whole line will be displayed, including the right portion that is truncated on long commands while the program is listed on the screen.

You will be prompted to replace the command, delete the line or edit the parts of the command. If you choose to replace the command, you will be prompted for a new command through a series of bar menus. Use the arrow keys to move between the menus and to highlight commands. If you choose to delete the line, it will be deleted and the next program line displayed in its place. There is no confirmation prompt for deleting in this mode.

The default is always edit the active line. When you select edit, a new window will appear and prompts will appear to get the detail information for the command.

While in Add line mode, the menus of the different TAS-Plus command categories are displayed. Use the arrow keys to move between the menus and to highlight commands.

A description of the prompts for each command appears in Chapter VIII of the reference section. Note that the documentation page number for the command you are working on appears in the upper right corner of the screen. This provides a means for fast access to the information you need.

When you have finished with the prompts for a command, you will be asked if all of the above is correct. If you answer no, TAS-Plus will allow you to reenter the information form the top. If you answer yes, the new or changed command line will be added to your program. A line added while in Add line mode will appear directly after the active line and will become the new active line.

If you are in Add line mode, you will be shown the command menus to select another command to add. If you are in Change line mode, the pointer will move to the next line and give you the chance to edit it.

To return to the main editor screen from either Add line or Change line modes, press the Esc key. The program listing will reappear. To bring the Activity window up, press Esc again.

DELETE LINE

You may select this activity if you want to delete the active line. You will be prompted for confirmation before the line is deleted.

INSERT ON/OFF

This option changes between Insert on and Insert off, depending on whether you are in Change line mode or Add line mde. It is used to toggle between Change and Add.

SCREEN ADD/CHG

TAS-Plus programs may have up to 16 screen and report format screens. To create a new screen or report format, select this option. You will see a list of current screens and reports currently associated with this program, if there are any. There is always an

option for new screen also. Upon selecting a screen, you will be asked if it is a screen or report. The screen painter/ report formatter will then appear. For more information on screen painting, see the next chapter.

DEL SCREEN

This selection is used to delete a screen or report format from a program. You will be shown a list from which to choose the screen or report to delete.

PRINT PROGRAM

You can enter a beginning and ending line number to print, or press return to select from the first line to the last line. You may print the program listing to the screen, printer or disk. You will be prompted for the destination of the program printout.

Two sets of line numbers will be printed. The first one is the actual line number. If you get an error during compiling, this is the line number you should use to determine the problem line. The second line number is the run line number. It is the number of the line after compiling. During compiling, certain program lines are skipped. This decreases the size of the compiled program and makes it run faster. If you get a run-time error message, use the the run line number to find the problem line.

If you are printing to the screen, follow the prompts to press any key to continue as the program listing is presented screen by screen.

COMPILE PROGRAM

This selection will let you compile the current program with out exiting the source code editor. You must save the program first if you have made changes or the changes will be lost. After compiling, you will return to the source code editor. You may either load the program again or press Esc to return to the TAS-Plus Main Menu.

SAVE PROGRAM

This choice is used to save the program you are editing. After it is saved you have the option of compiling or returning to the TAS-Plus editor. You may either load the program again or press Esc to return to the TAS-Plus Main Menu.

FIELD CHK/ADD

You may look at the existing field descriptions or add new field descriptions to the data dictionary with this selection. You will be prompted for the field specifications.

GOTO LABEL/LINE

This is a fast way to jump to a particular line label or line number. If you give a line number that is higher than the last line of the program or a line label that doesn't exist, the pointer will stay at the currently active line.

EXIT

This selection provides an exit from the editor to the Main Menu. If you have made any changes to the program since the last time you saved, you will be warned that the changes will be lost if you continue. You may then select to continue and lose the changes or to cancel the exit so that you can save the program. If you choose to exit, the program listing will be cleared and you will be given the chance to enter a new program name to edit. If you press Esc at the program name prompt, you will return to the TAS-Plus Main Menu. Pressing return without a filename at that prompt will have the same effect as pressing Esc.

VI. Screen Painter

The TAS-Plus screen painter can be accessed through the Create program selection and the Edit program selection from the Main Menu. When you choose Create program you proceed directly into the screen painter. Through the source code editor, you press Esc to bring up the activity window and select Screen add/chg.

Information about the screen painter will be displayed on the screen. Follow the prompts to move through the two screens and read the information. The screen will blank except for the screen title and cursor position on the bottom line. The screen is now a drawing board for you to use however you want.

Use the arrow keys to move the cursor on the screen. You can always tell what column and row you are in by the position indicators on the bottom line. You may type any text you want in any position on the screen. Beyond just typing text, there are other options that make the screen painter so versatile that you can produce anything the hardware is capable of.

When accessing the screen painter from the source code editor, you can paint two types of screens. One type is the standard display screen that is used for viewing and entering data on the monitor. The maximum size for a standard screen is 80 characters wide by 24 rows long. The other type is a report format which is used for printing database reports to the printer, screen or disk file. The Print report line command in the source code editor causes a particular line from a report format to be printed. A report format may be up to 128 characters wide and 60 lines long. You can print more than one report format line on a physical line by using the no carriage return option on the first Print report line command, up to a limit of 255 characters on a line.

Also, a field may only appear once on a standard display screen, but may appear any number of times on a report format. A report format may not be used for data entry, only for data printing or display.

To see the screen choices, press the Esc and a command window will appear on the screen. Each choice with a description is listed below.

ADD FIELD ON SCREEN

This choice will put a field of your choosing on the screen with the left-most character of the field at the cursor position. If the field is present in the data dictionary or defined in the current program, reverse video representation of the field will appear on the screen. If the field isn't present in the data dictionary, you will be asked if you would like to add it. If you add the field you will be prompted for the information TAS-Plus needs to put the field in the dictionary.

REMOVE FIELD

This option will remove a field from the screen. The cursor must be in the left-most position of the field to remove it. After removing it, you may place it elsewhere on the screen.

SHOW FIELD INFO

The field description from the data dictionary can be displayed with this choice. The cursor must be in the first position of the field. The description is for information only, changes cannot be made. To make changes to a field description, save the screen and use the Field chk/add selection on the editor activity window.

JUMP TO LOCATION

This choice lets you move the cursor directly to a particular column and row without using the arrow keys.

BOX

The box selection will draw a box on the screen for you. The type of box (single line, double line, etc.) may be set in the installation program. The default is a double-line box.

To draw a box, put the cursor in the position you want for the lower left corner of the box and choose Box from the menu. A message will display telling you to move the cursor to the position you want for the upper-right corner of the box and press return. Press any key to remove the message from the screen and then move the cursor as directed. Be sure to use only the arrow keys to move the cursor after the Box command is selected. You will not be allowed to move the cursor farther left or farther down than your starting point. When the cursor is in the position you want for the upper-right corner of the box, press return. The box will appear on the screen and the cursor will return to the starting point.

A straight line up and down may be drawn by moving the cursor to a row position without changing the column position. A line left and right can be drawn by moving the cursor to a column position without changing the row position.

To remove a box, or portion of a box, use the space bar to blank it out.

PRINT SCRN

This selection will make a hard copy of the screen on your printer. Make sure that the printer is on-line and the paper is positioned properly. If your printer can print graphics characters, you will get an accurate representation of what is on the screen, excepting highlighting and color. If your printer does not directly interpret the IBM-PC graphics character set then you will not get the graphics printed properly.

COLOR CHOICES

The color choices selection is used for both highlighting and color. It will present a menu with different background color choices, a menu for foreground color choices and a prompt to activate flashing. Once your selection is made, that color combination will be used wherever you move the cursor, just like it was a paintbrush. If you have selected a graphic, it will be displayed in the color combination you have chosen. To turn off the color painting, press the Ins key on the numeric pad. This tells the program to leave whatever is on the screen so you can move the cursor without painting on the screen. You can toggle your last selection back on by pressing the Ins key again. To erase a color that has been painted on the screen, choose another color combination and move over the area erase with the arrow keys. For more information on highlighting and color, please see Appendix XI.

GRAPHIC CHARS

All 159 characters of IBM's graphic set is available to use with this selection. Simply choose the graphic character you want from the menu displayed with this choice. The character you select will be printed on the screen every time you press an arrow key. It will display in the color you have active at the time. To turn off the graphic, press the Del key. You will then be able to move the cursor with the arrow keys without painting the graphic. You can toggle the last selection back on by pressing the Del key again

EXIT SCRN PAINTER

Use this selection to save your screen or exit the screen painter without saving it. If you have used any of the color choices to change highlighting or color then the screen saved will be a minimum size of 3,840 bytes. If you don't use the color selection then the typical screen is 500 to 1,000 bytes in size.

If you are accessing the screen painter from Create program, you must have at least one data field on on the screen to save it. Your

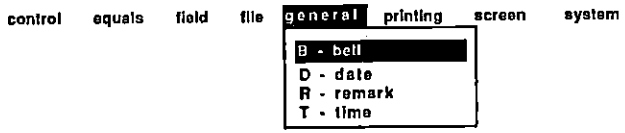
screen will be turned into a source program, compiled and you will be given the options of entering data or returning to the TAS-Plus Main Menu.

If you have accessed the screen painter from the source code editor, you will return to your program listing.

VII. List of TAS-Plus commands

The following are a list of the commands available in the TAS-Plus language. They are in alphabetical order. The command name is first on the page followed by a picture of the choice you would make in the editor to use this command. Also included are the definitions of the parts of the command and examples.

bell



The **bell** command is used to send a short sound out the terminal speaker.

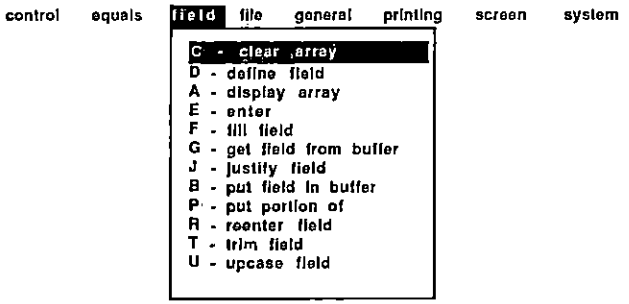
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

sound bell

clear array



The **clear array** command is used to clear a memory array field. Use the **screen lock/unlock** command to redisplay the field after clearing .

Note: This could also be accomplished using a **for/next** command and an **equals** command, however **clear array** is much faster.

PARTS OF THE COMMAND:

mem field - This is the name of the memory array field to be cleared. If you enter just the name without a subscript *[ss]* , then the program will clear from the beginning of the array field (as though you had entered a subscript of 1). You can, however, enter an array element number or field so that the field is in the standard format of *fieldname[ss]* and the clearing will start with that array element number. This is required.

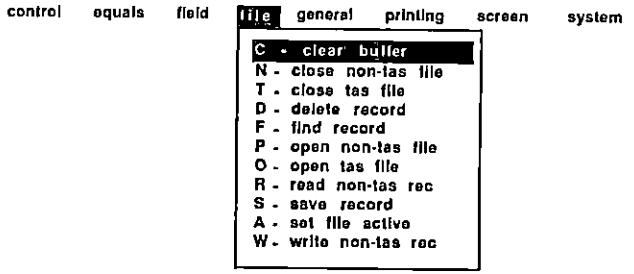
number of elements - This is a numeric constant which tells how many array elements are to be cleared. For example, if the field has 10 array elements and you want to clear six of them, then this value would be 6. This is required.

TAS EDITOR EXAMPLE:

clear array: field: TESTARRAY # of elements: 10

clear array: field: TESTARRAY[5] # of elements: 5

clear buffer



The **clear buffer** command is used to clear the record buffer and/or record number of a file. If the record number is cleared (set to 0), then the next record saved will be treated as a new record.

Note: If the record buffer is cleared, the record number will also be cleared. However, you can clear the record number and not the buffer. This allows parts of the previously saved record to be used as defaults in the new record.

PARTS OF THE COMMAND:

filename - This is the name of the file to be cleared.

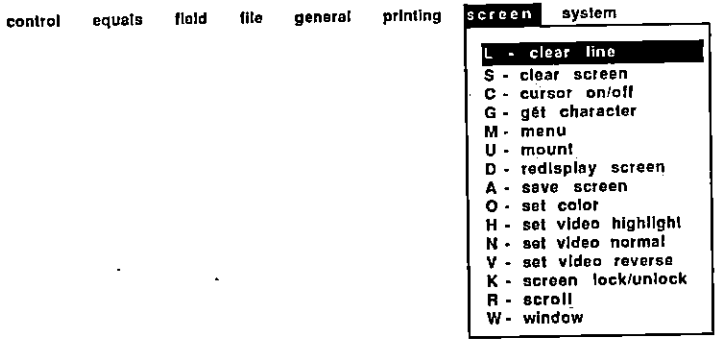
what to clear (RB) - R - clears the record number and B - clears the buffer.

TAS EDITOR EXAMPLE:

clear buffer: TEST RECORD NUMBER ONLY

clear buffer: TEST RECORD NUMBER AND BUFFER

clear line



The **clear line** command is used to clear a line of text displayed on the terminal screen.

PARTS OF THE COMMAND:

column - This is the column location of the line to be cleared. This may be a numeric constant or a field in the format *fieldname [ss]*.

row - The row location of the line to be cleared. May be a numeric constant or a field in the format *fieldname [ss]*.

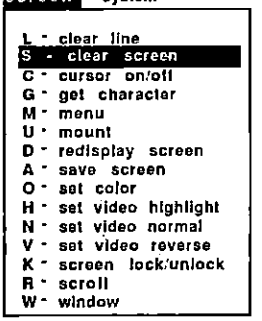
Note: If the location is not given, then the default is the last cursor location.

TAS EDITOR EXAMPLE:

clear line: 1 row: 10

clear screen

control equals field file general printing **screen** system



The **clear screen** command is used to clear the terminal screen of all text.

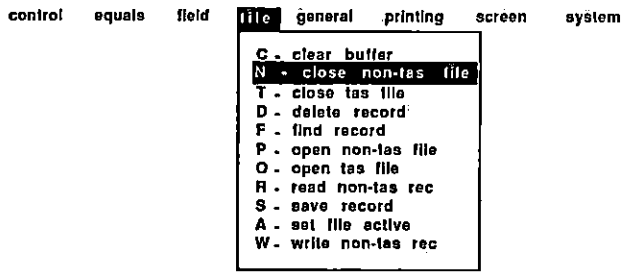
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

clear screen

close non-tas file



The **close non-tas file** command is used to close a file (other than standard TAS data file) that was previously opened with the **open non-tas file** command.

Note: Only one non TAS file may be open at any time. This command must be used before attempting to open another non TAS file. To assure that data is saved, the file should be closed before exiting.

PARTS OF THE COMMAND:

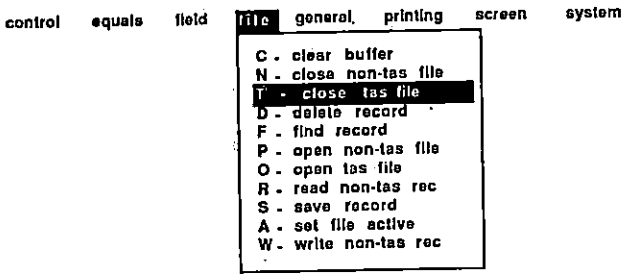
delete after close (YN) - This will inform TAS whether or not to delete the file after it is closed. The default is N.

TAS EDITOR EXAMPLE:

close non-tas file: DELETE AFTER CLOSE

close non-tas file:

close tas file



The **close tas file** command is used to close a standard TAS data file that was previouslyopened using the **open tas file** command in the same program.

Note: The **close tas file** command isn't normally necessary since TAS files are automatically closed when the program is exited.

PARTS OF THE COMMAND:

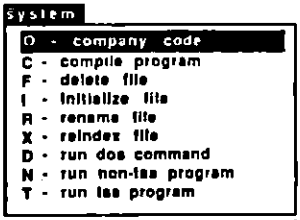
filename - This is the name of the file to be closed. This is required.

TAS EDITOR EXAMPLE:

close filename: TEST

company code

control equate field file general printing screen system


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **company code** command is used to set the TAS internal company code from a field or to get the current company code and put it in a field without having to return to the main menu.

PARTS OF THE COMMAND:

fieldname - This is any legal field using the standard format of *field name[ss]*. This is required.

get or set - G for get internal company code, S for set internal company code. This is required.

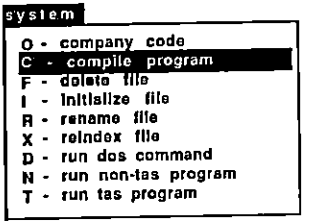
TAS EDITOR EXAMPLE:

company code: field: CO.CODE GET

company code: field: CO.CODE SET

compile program

control equals field file general printing screen system


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **compile program** command is used to compile the TAS program specified. You may send the results to the screen, printer or to a disk file. The program will return to the next line after the compiler is finished.

PARTS OF THE COMMAND:

filename - This is the name of the file to be compiled. It can either be an alphanumeric constant (in quotes) or an alphanumeric field. The extension must be included. (Example: .SRC or .EDT). The file name may be preceeded by a pathname. This is a sequence of directory names, each separated from the previous one by a backslash and ending with a backslash. The pathname can be preceeded by a drive designator. An example might be C:\DIRECTORY\SUBDIRECTORY\.

print results to - This is where compiler error messages will be sent. Choices are: A - Ask at run time, S - Screen, P - Printer, D - Disk file, and H - Hide results (no print). If you choose "Disk file", it will ask you for a disk file path and name.

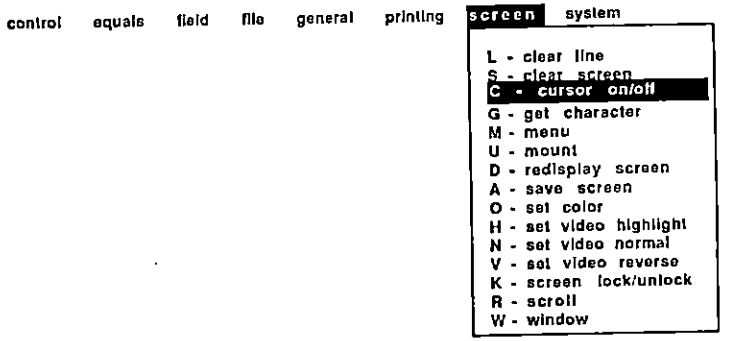
TAS EDITOR EXAMPLE:

compile prg: TESTPROG.EDT print results to: S

compile prg: \MYDIR\TESTPROG.SRC print results to: P

compile prg: TESTPROG.EDT print results to: D

cursor on/off



The **cursor on/off** command is used to turn the cursor on and off. This command is used when you don't want the cursor to appear on the screen for any reason.

PARTS OF THE COMMAND:

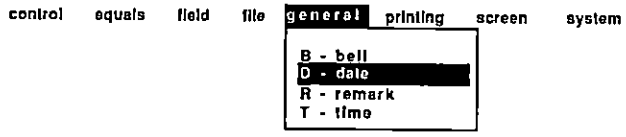
cursor (YN) - Y to turn the cursor on, N to turn the cursor off.

TAS EDITOR EXAMPLE:

turn cursor: ON

turn cursor: OFF

date



The **date** command is used to set the system date from a field or to get the date from the system and put it in a field.

PARTS OF THE COMMAND:

date field name: This is the field that will receive the date or set the date. It is a field of type D or type S using the standard format of *fieldname[ss]*.

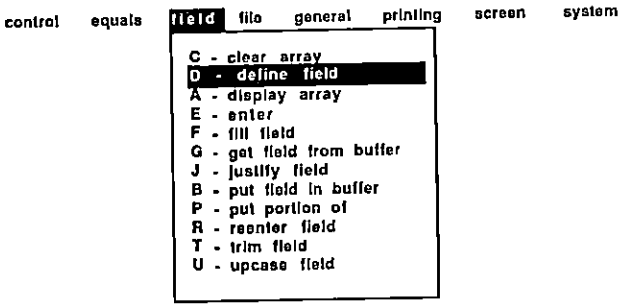
get or set - G for get system date, S for set system date.

TAS EDITOR EXAMPLE:

system date: field: TEST GET

system date: field: TEST SET

define field



The **define field** command is used to create a temporary variable that will effect the current program only.

PARTS OF THE COMMAND:

fieldname - This is the name you assign to the temporary variable. It should be different from all other fieldnames in the program. This is required.

type - These are the field types available:
A - alphanumeric - field will accept up to 255 alpha characters or numeric characters.
N - numeric - field will accept up to 20 numeric characters with an even number of decimal characters. If total field lenght is an odd number of characters, the value may contain negative. Even numbered field lenghts do not allow a negative value. Negative values are displayed with a minus sign (-) preceding the number.
D - long date format - field is 10 characters long. A space or zero may precede a single digit day or month in the format MM/DD/YYYY.
O - overlay - field combines two or more fields. To use, place command immediately before the

fields that are to be overlayed and set the length equal to the combined lenghts of fields.
S - short date format - field is 8 characters long in the format MM/DD/YY.
T - time field - field is 8 characters long in format HH:MM:SS.
This field is required.

size/filename - This is the number of characters in the field. The above rules apply to field size. If the field type is set to R, then instead of asking for field size, it will ask for the filename that the field will pertain to. This field is required.

decimal characters - This is the number of decimal characters in a numeric field. It must be an even number (i.e. 0, 2,4,6 or 8).

array elements - This is the number of elements if this is an array. The maximum number of elements is 255.

NOTE: You must make sure that the last **DEFINE**'d field in a program is NOT an array. If it is the program will not run properly. Data will appear to change without you doing anything to change it.

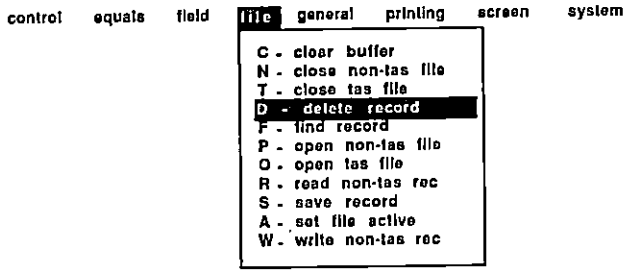
TAS EDITOR EXAMPLE:

define: TEST type: A size: 25

define: TEST type: N size: 9 decimal: 2

define: TEST type: S size: 8

delete record



The **delete record** command is used to delete a single record or all the records in a file.

Note: After using the **delete record** command, you must do a Find-Begin, Find-End, Find-Generic or Find Match before using **delete record** command again. A Find-Next or Find Previous will cause a file error.

PARTS OF THE COMMAND:

filename - This is the filename where the record or records to be deleted are located. This is required.

confirm (Y/N) - Y will ask at runtime whether or not to delete the record. N - will delete the record without asking. The default is Y.

delete all records - Y will delete all of the records in the file. N will delete a single record. The default is N.

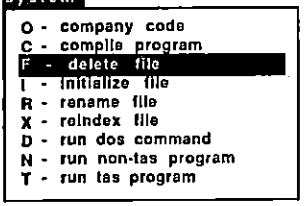
goto label - This tells the program where to go if you choose not to delete a record at runtime. This only works if the confirm option above was answered Y.

TAS EDITOR EXAMPLE:

```
delete record: TEST confirm: Y all recs: N goto
label: LABEL
```

delete file

control equals field file general printing screen system


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **delete file** command is used to delete a file on the disk. You may use path names and MS-DOS wildcards.

PARTS OF THE COMMAND:

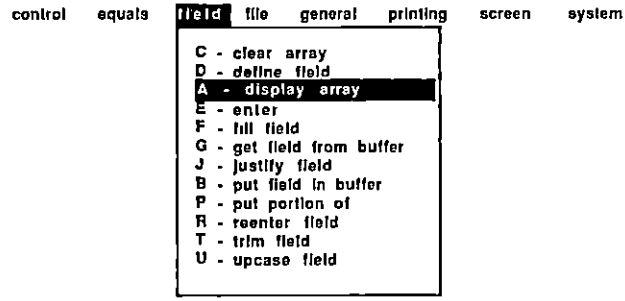
filename - This is the name of the file to be deleted. It may be a field. The extension must be included. (Example: TEST.SRC or BTGLR.K). The filename may be preceeded with a pathname. This is a sequence of directory names, each separated from the previous one by a backslash and ending with a backslash. It may be a field. The pathname can be preceded by a drive designator. An example might be C:\DIRECTORY\SUBDIRECTORY\.

TAS EDITOR EXAMPLE:

delete file: TEST.SRC

delete file: \MYDIR\BTGLR.K

display array



The **display array** command is used to display an entire alpha array or portion of an alpha array on screen as though it were a single block. Each element of the array would normally appear as a different line.

PARTS OF THE COMMAND:

field name - The array field to be displayed. Must be alphanumeric array field.

offset - The position number within each array element to start displaying. For example, Position 1 is the first character in each array element, whereas position 10 is the 10th. Any legal field or numeric constant.

column - What column to start the printing. Any legal field or numeric constant.

row - What row to start the printing. Any legal field or numeric constant.

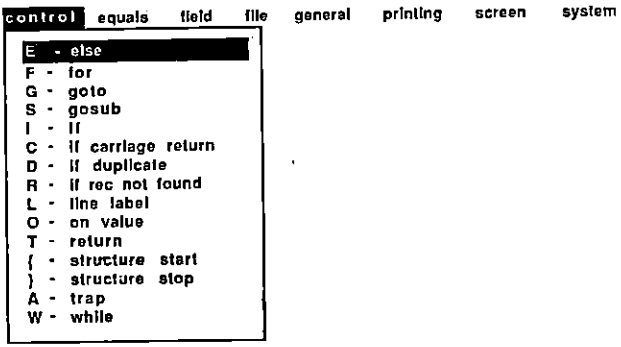
width - How many characters wide to print. Any legal field or numeric constant.

number of lines - How many lines or rows to print Each line would be a different array element. Any legal field or numeric constant.

TAS EDITOR EXAMPLE:

display array: fieldname: ARRAY.FLD offset: 1 col: COL.NUM row: ROW.NUM width: WDTH.NUM lines: NUM.LNES

else



The **else** command is used in conjuction with an **if** command as part of a command structure. The **if** command is in the form of a true/false test. If the test is true, then the program will go to the **structure start** on the next line. The commands following will be executed until the next corresponding **structure stop** is found.

However, if the test result is false, then the program will jump to the **else** command followed by an **structure start** on the next line. The commands following will then be executed until the next corresponding **structure stop** is found.

Note: For a more complete explanation of the **else** command, see section XII, Advanced TAS-Plus topics in the tutorial.

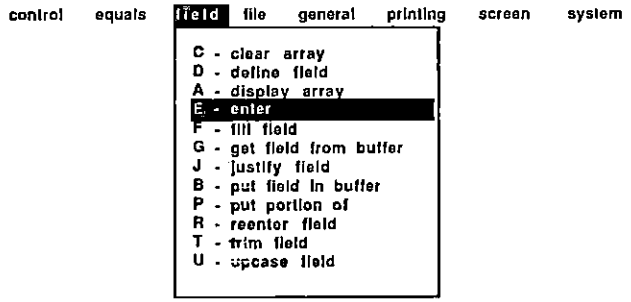
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

else

enter



The **enter** command is used to enter data into a field on a screen. The command automatically knows the location of the field (from the screen format) and the type, size, etc. (from the data dictionary or the define within the program).

PARTS OF THE COMMAND:

fieldname - This is the name of the field to be entered. It must be in the data dictionary or a defined field within the program in the format *fieldname[ss]*. It may be an element in an array. This is required.

mask string - This is a set of characters that are acceptable as input. For example, if Y and N are the only characters that you want to be entered in a field, then the mask would be "YN". If the mask is a field name, then no quote marks are required. If the mask is a string, then it must have quotes around it.

help label - This is where program control will be transferred if help key is entered . This acts like a **gosub** and you should return with the **return** command.

up arrow label - This is where program control will be transferred if the up arrow is entered. If this choice is left blank, then the global setting will control, or cursor will move to previous **enter** command.

upcase (YN) - Y - all the characters entered using this entry command will have their case changed to upper. N - the case is left as it was entered. The default is N.

auto cr (YN) - Y - the program will automatically do a carriage return after the operator has entered the last character in the field. N - the cursor will remain in the field until operator manually enters a carriage return. (This would allow changes to the field while entering before preceding to the next field). The default is N.

password type - Y - the screen will display questions marks instead of what is being input on the keyboard. N - will display keyboard input.

enter at column number - This allows entry to be made at a specific place on the screen. This will be the column number of the left most character. If a specific location is not entered, the field must be a part of the most recently mounted screen.

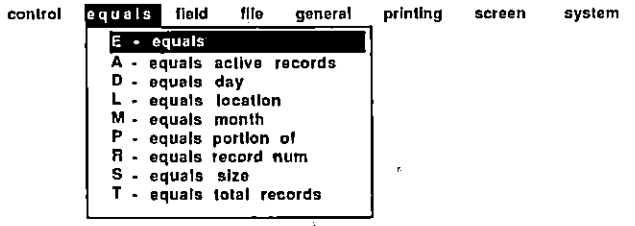
enter at row number - This will be the row number of the left most character. If this is left blank, then entry will default to position defined in screen setup. If a specific location is not entered, the field must be a part of the most recently mounted screen.

TAS EDITOR EXAMPLE:

enter: TEST mask: "YN" help: HELP.LABEL up
arrow: PREV.LINE PASSWORD

enter: TEST cloumn: 30 row: 10

equals



The **equals** command is used for setting one field equal to the value of a constant or another field.

PARTS OF THE COMMAND:

receiving field - This is any legal field using the standard format of *fieldname[ss]*. This is required.

expression - The expression may be a single field, several fields, or a combination of fields and constants. Parts of the expression may be surrounded by parentheses to force the program to evaluate those portions first. The first field to be evaluated in the expression will determine whether or not the expression is treated as alphanumeric or numeric type. If the expression is determined to be alphanumeric then all fields are added (concatenated) together. If the expression is numeric then the legal operators are addition (+), subtraction (-), multiplication (*) and division (/).

Note: Operators are evaluated in the order in which they appear in the expression. That is, muliplication and division do not have precedence over addition and subtraction. If

nested parenthesis are used, it will start with the middle nest and work its way out.

Note: The **equals** command may also be used to convert one data type to another. See Reference Guide Section IX, How To Convert Data Types.

TAS EDITOR EXAMPLE:

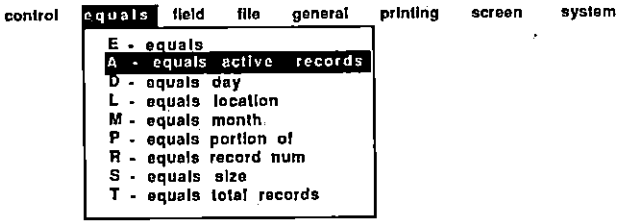
TEST equals: (NEWFIELD)

TEST equals: (OTHER.FIELD * 0.80)

TEST equals: (100)

TEST equals: ('ABC'+'DEF')

equals active records



The **equals active records** command is used to get the number of net records (total less deleted) in a particular file and place the numeric value into the receiving field.

PARTS OF THE COMMAND:

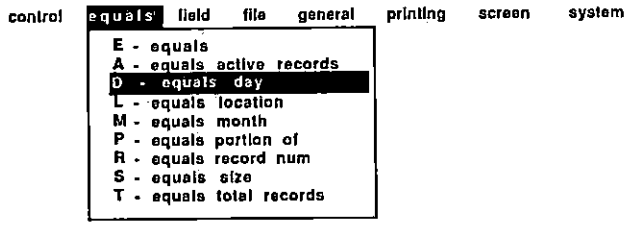
receiving field - This is any legal numeric field in the format of *fieldname[ss]*. This must be 8 characters in size with no decimal characters. This is required.

filename - This is the name of the file to be checked. This cannot be a field but must be the actual file name. This is required.

TAS EDITOR EXAMPLE

FIELD equals active # records: TEST

equals day



The **equals day** command is used to place the day number or name in the receiving field. If the receiving field is alphanumeric, the result will be the day name (MON, TUE, WED, THU, FRI, SAT, SUN). If the field is numeric, the result will be the day number (0-6). The expression must be a legal date field.

PARTS OF THE COMMAND:

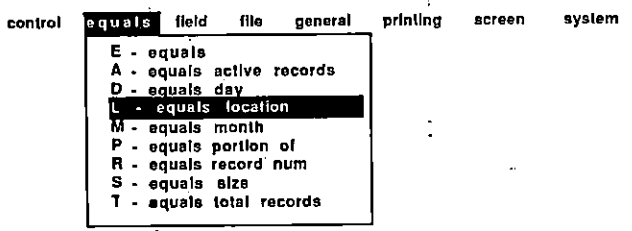
receiving field - This is any legal field using the standard format of *fieldname[ss]*. This is required.

date field - This is any standard date field. It may also be an element in an array.

TAS EDITOR EXAMPLE:

TESTDATE equals day number/name: DATE

equals location



The **equals location** command is used to determine the location of a group of characters within another field. For example, if you found the location of 'AB' within the field 'CDEFABQ,' the result in the receiving field would be 5. If the characters searched for are not within the larger group, then the result will be 0.

PARTS OF THE COMMAND:

receiving field - This is where the value will be placed. It is any legal numeric field using the standard format of *fieldname[ss]*. This is required.

chrs to search for - This is what you are searching for. It is an alphanumeric constant surrounded by quote marks (i.e. 'AB') or a legal alphanumeric field using the standard format of *fieldname[ss]*. This is required.

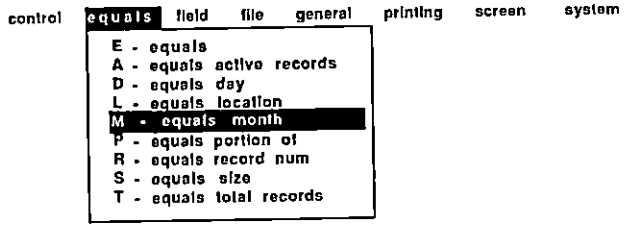
field to search within - This is the field you are searching in. It is any legal alphanumeric field using the standard format of *fieldname[ss]*. This is required.

starting character number - This lets you start searching at a specific character within the search field. It is a numeric constant or any legal numeric field using the standard format of *fieldname[ss]*. The default is 1.

TAS EDITOR EXAMPLE:

TEST equals location of: "AB" within: TEST2

equals month



The **equals month** command is used to place the month number or name in the receiving field. If the receiving field is alphanumeric, the result will be the month name (JAN, FEB, MAR, APR, MAY, JUN, JLY, AUG, SEP, OCT, NOV, DEC). If the field is numeric, the result will be the month number (1-12). The expression must be a legal date field.

PARTS OF THE COMMAND:

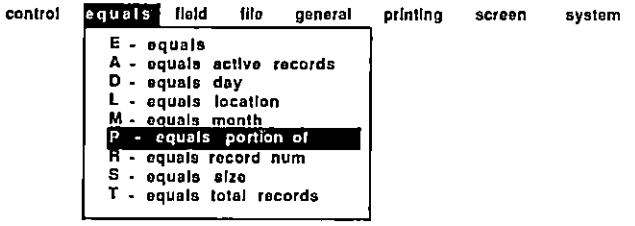
receiving field - This is any legal field using the standard format of *fieldname[ss]*. This is required.

date field - This is any standard date field. It may also be an element in an array. This is required.

TAS EDITOR EXAMPLE:

TEST equals month number/name: MONTH

equals portion of



The **equals portion of** command is used to take a portion of a field and place it the receiving field. The receiving field must be alphanumeric. If the receiving field is not large enough, then the use field will be truncated (on the right) to fit the receiving field.

PARTS OF THE COMMAND:

receiving field - This is any legal alpha-numeric field using the standard format of *fieldname[ss].* This is required.

use field - This is the field that a portion will be taken from. This will not effect this field at all but will just make a copy of the characters. It is any legal alphanumeric field using the standard format of *fieldname[ss].* This is required.

starting character number - This lets you start copying at a specific character within the search field. It is a numeric constant or any legal numeric field using the standard format of *fieldname[ss]* . The default is 1.

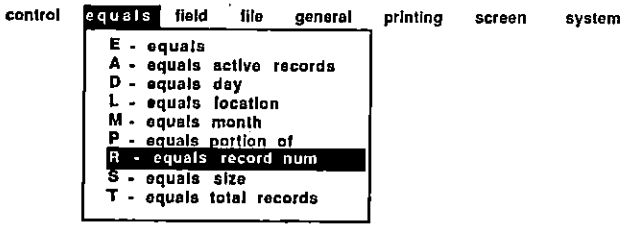
number of chrs to move - This is the actual number of characters that will be copied into the receiving field. This is required.

TAS EDITOR EXAMPLE:

TEST equals portion of: PARSEDFIELD start: 1
num chrs: 15

TEST equals portion of: PARSEDFIELD start:
STARTNUM num chrs: LENGHT

equals record num



The **equals record num** command is used to move the current record number of a file into the receiving field or to move the value in the receiving field into the file record number. In the case of the latter, the record corresponding to the value moved will also be found.

Note: If the record for the value specified has been previously deleted, the program will attempt to find the next undeleted record. If the end of the file is reached before another undeleted record is found, a file error will occur. If an attempt is made to read a record beyond the total records in the file, a file error will occur.

PARTS OF THE COMMAND:

receiving field - This is any field defined as type R.

filename - This is the actual name of the file to be checked. It cannot be a field. No surrounding quotes are required. This is required.

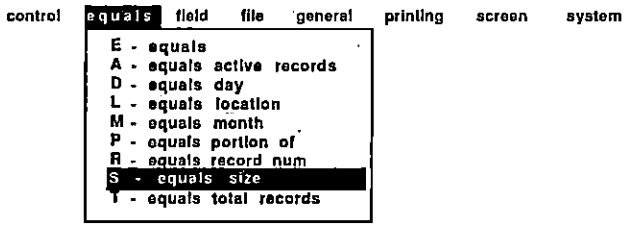
get or set - G - will get the value of the current record number and place it in the receiving field. P - will put the value of the receiving field into the record number and an attempt to read the record will be made.

TAS EDITOR EXAMPLE:

FIELD equals record num: TEST get:

FIELD equals record num: TEST set:

equals size



The **equals size** command is used to place the length (defined or actual) of an alpha type field into a receiving field.

PARTS OF THE COMMAND:

receiving field - This is the field into which the size value will be placed. It is any legal numeric field using the standard format of *fieldname[ss].*

which type (AD) - A - is for actual field size, D - is for defined field size.

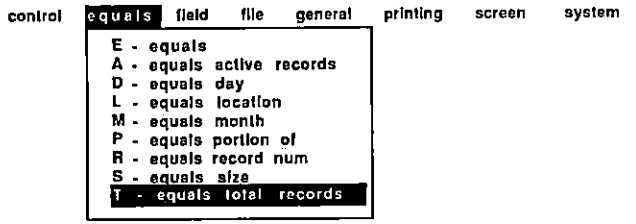
field being checked - This is the field you want the size of. It is any legal alphanumeric field using the standard format of *fieldname[ss].*

TAS EDITOR EXAMPLE

TEST equals size: ACTUAL of field: FIELD1

TEST equals size: DEFINED of field: FIELD1

equals total records



The **equals total records** command is used to get the number of total records in a particular file and move the number into the receiving field. This will include previously deleted records. See **equals active record** command to get number exluding deleted records.

PARTS OF THE COMMAND:

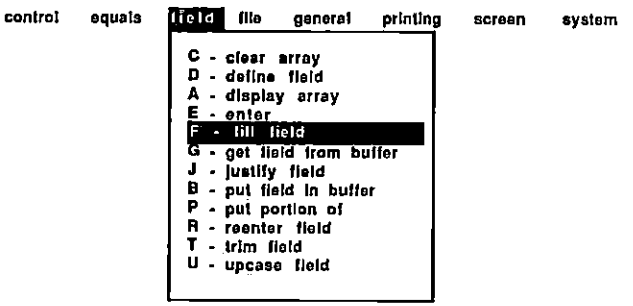
receiving field - This field must be 8 characters in size with no decimal characters. It is any legal numeric field in the format of *fieldname[ss]*. This is required.

filename - This is the actual name of the file to be checked. It cannot be a field. No surrounding quotes are required. This is required.

TAS EDITOR EXAMPLE:

TEST equals total # records: TEST

fill field



The **fill field** command is used to fill an alphanumeric field with a certain character. For example, if you wanted a field to contain a dollar amount preceded by a dollar signs, you would first move a numeric amount to an alphanumeric field and justify it to the right. Then you would use the **fill field** command to fill the leading spaces with $.

PARTS OF THE COMMAND:

field to be filled - This is any legal alphanumeric field in the format of *fieldname[ss]*. This is required.

fill (LT) - L - will fill the leading spaces with characters. T - will fill the trailing spaces with characters. The default is L. This is required.

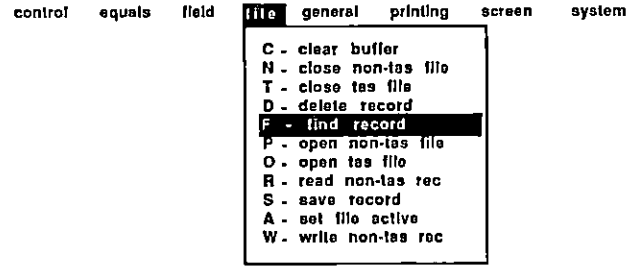
character to be used - This is the character to be used as the filler. This is required.

TAS EDITOR EXAMPLE:

fill field: TEST LEADING with: $

fill field: TEST TRAILING with: #

find record



The **find record** command is used to find a specific record in a file. The record searched for must have at least one key. The field used to search by must must be designated as a key field in the data dictionary.

Note: The find related key option of this command forms the basis for much of the relational aspect of TAS-Plus.

PARTS OF THE COMMAND:

find type - This is the type of the search to be done. The following choices are available:
M - match or exact search - This searches for the first exact match. If the search goes beyond the last value in the file, control is transferred to the on-error line label.
G - generic search - This searches for the first match of a partial key or the next record if a match is not found. If the search goes beyond the last value in the file, control is transferred to the on-error line label.
B - beginning record - This searches for the first record in the file.
E - last or ending record - This searches for the last record in the file.

N - next record - This searches for the next record in the file.
P - previous record - This searches for the previous record in the file.
R - related record - This searches for a related field in another file. If an exact match is not found, control is transferred to the on-error line label.

fieldname/filename - If the search is N or P, then this is the filename. Otherwise, this is the field that will be used for the search. It must be designated as a key field in the data dictionary. It can not be an array field.

related fieldname - If the search is a type R, then this field contains the value you are searching for. It must be designated as a key field in the data dictionary. It can not be an array field.

on error label - This is the line label where control will be transferred if a file error occurs during the search or record read. If an error label is not included and an error occurs, then the error message will be displayed on the bottom of the screen and the program will continue executing with the next line.

TAS EDITOR EXAMPLE:

find: MATCH EXACT using field: TEST on error goto: LABEL

find: GENERIC using field: TEST on error goto LABEL

find: BEGINNING using field: TEST

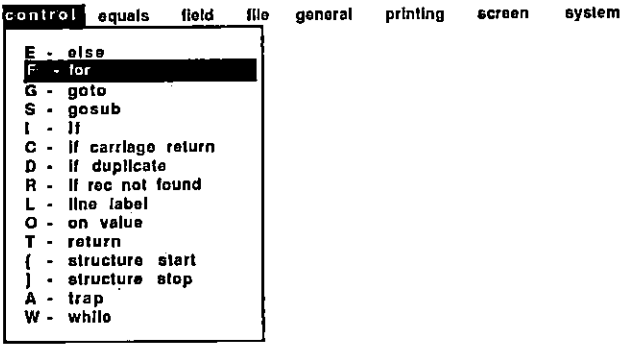
find: ENDING using field: TEST

find: NEXT in file: TEST

find: PREVIOUS in file: TEST or error goto: LABEL

find: MATCH EXACT using field: TEST1 USING RELATED FIELD TEST2 or error goto: LABEL

for/next



The **for/next** command is used as the first part of a loop. The beginning and ending portion of the loop are **structure start** and **structure stop**. Each **structure start** and **structure stop** command is on its own line. A loop will always be executed at least once.

Note: For a more complete explanation of the **for/next** command, see section XII, Advanced TAS-Plus topics in the tutorial.

PARTS OF THE COMMAND:

counter - This is the field that is increased or decreased each time the loop is run. This is any legal numeric field in the format of *fieldname[ss]*. The counter size must be at least the same size as your start and stop variables.

Note: An easy mistake to make is to use values for the start and stop fields of 1 and 10 and to make the counter variable 1 character in length. If that were to happen, then the counter would never reach 10 (the 10 would become 0 since only 1 character would be allowed). This is required.

start- This is the beginning number of the loop. This field may be a numeric constant or a legal numeric field in the format of *fieldname[ss].* Only integers (whole numbers) are allowed. This is required.

stop - This is the ending number of the loop. This field may be a numeric constant or a legal numeric field in the format of *fieldname[ss].* Only integers (whole numbers) are allowed. This is required.

step - This is the numeric value to advance the counter each time the beginning of the loop is encountered. The value may be a numeric constantant or any legal numeric field in the format of *fieldname[ss].* If the value is negative (less than 0), then the amount is subtracted from the counter. If the value is positive, it is added. The default is 1. This is required.

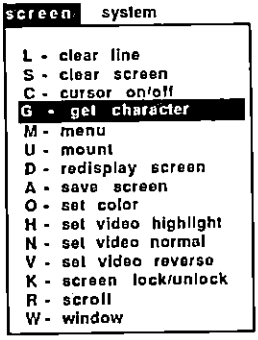
TAS EDITOR EXAMPLE:

for/next: counter field: TEST start: 1 stop: 10 ste p: 1

for/next: counter field: TEST start: 100 stop: 50 step: -1

get character

control equals field file general printing **screen** system



The **get character** command is used to get or set the character and screen atribute which is currently on the screen at the specified row and column location.

PARTS OF THE COMMAND:

column value - This is the column where the character is located. It may be a numeric constant or a numeric field in the format of *fieldname[ss].*

row value - This is the row where the character is located. It may be a numeric constant or a numeric field in the format of *fieldname[ss].*

character field - This is the field that will get or set the actual printable character on the screen. It may be any alphanumeric field in the format of *fieldname[ss*].

atribute field - Every character on the screen may have its own atribute value. This determines what color and intensity the character will be displayed with. This field will be used to either get the current value or set the current location to the value of the field. For example, on monochrome monitors 7 is the value of the normal intensity white on black screen atribute. It

may be any numeric field in the format of *fieldname[ss]*.

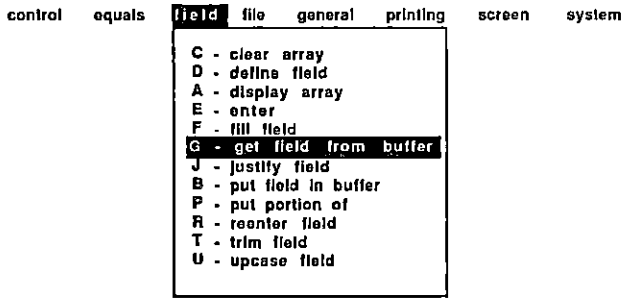
get or set chr and atr - This choice will determine whether you want to get or set the character and atribute. G - get / S - set.

TAS EDITOR EXAMPLE:

get/set chr & atr: column: 1 row: 1 chr fld: SCREENFIELD atr fld: ATRIBUTE.FLD GET

get/set chr & atr: column: X row: Y chr fld: NEWFIELD atr fld: CUR.ATR SET

get field from buffer



The **get field from buffer** command is used to get a non-tas field (that is specified by fieldnumber) and place it into the TAS alpha type receiving field. This command works in coordination with the **read non tas rec** command. The non-tas record to be read would be part of a file (quote/comma delimited) previously opened with an **open non-tas file** command. For example, field 3 in the record A,B,C,DEF,102 would be C.

PARTS OF THE COMMAND:

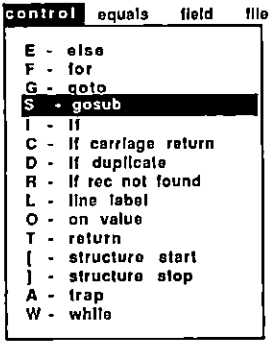
receiving field - This is the tas field that will receive the data. It is any legal alphanumeric field using the standard format of *fieldname[ss].* This is required.

delimited field number - This is the specific field within the non-tas file that is to be read. It is any legal numeric constant or numeric field using the standard format of *fieldname[ss].* This is required.

TAS EDITOR EXAMPLE:

TASFIELD equals delimited field #: 3

gosub



The **gosub** command is used to transfer control of the program to the line label specified. When the program finishes the subroutine, it should encounter a **return** command. Control will then be transferred back to the next line of the original program.

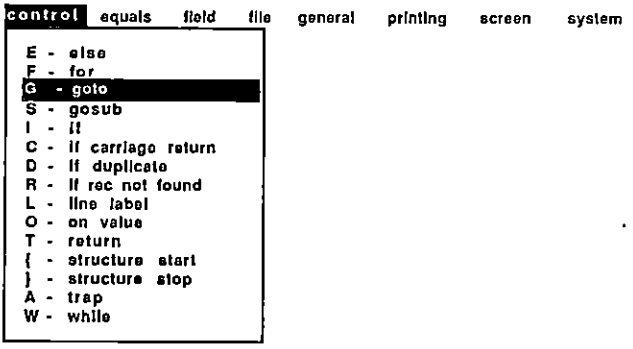
PARTS OF THE COMMAND:

line label - This is the label of the line to which control will be transferred. A line label can be up to 10 character long. It can only contain letters and numbers. Quote marks, math and other symbols are not allowed.

TAS EDITOR EXAMPLE:

gosub: LABEL

goto



The **goto** command is similar to the **gosub** command in that it also transfers control to a specific line label. However, with the **goto** command the transfer is absolute. The only way the program can return to this location is with another **goto**.

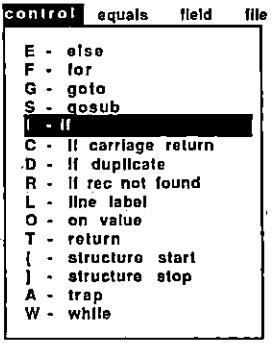
PARTS OF THE COMMAND:

line label - This is the label of the line to which control will be transferred. A line label can be up to 10 character long. It can only contain letters and numbers. Quote marks, math and other symbols are not allowed.

TAS EDITOR EXAMPLE:

goto: LABEL

if



The **if** command is used in conjuction with an **else** command as part of a command structure. The **if** command is in the form of a true/false test. If the test is true, then the program will go to the **structure start** on the next line. The commands following will be executed until the next corresponding **structure stop** is found.

However, if the test result is false, then the program will jump to the **else** command followed by an **structure start** on the next line. The commands following will then be executed until the next corresponding **structure stop** is found.

Note: For a more complete explanation of the **if** command, see section XII, Advanced TAS-Plus topics in the tutorial.

PARTS OF THE COMMAND:

compared field - This is a legal field name in the format of *fieldname[ss]*. It may be alphanumeric, numeric, or date. If it is a defined field, it may be record or overlay type.

< / <= / = / => / > / <> - The comparison type must be one of these choices.

expression - The expression may be a single field, several fields, or a combination of fields and constants. Parts of the expression may be surrounded by parentheses to force the program to evaluate those portions first. The first field to be evaluated in the expression will determine whether or not the expression is treated as alphanumeric or numeric type. If the expression is determined to be alphanumeric, then all fields are added (concatenated) together. If the expression is numeric then the legal operators are addition (+), subtraction (-), multiplication (*) and division (/).

Note: Operators are evaluated in the order in which they appear in the expression. That is, muliplication and division do not have precedence over addition and subtraction. If nested parenthesis are used, it will start with the middle nest and work its way out.

goto label / gosub label / do / reent / ret / then - This determines where program control will be transferred. These are the choices:
GOTO LABEL is used to absolutely transfer control to the line label specified (just as if you had executed a standard **goto** command).
GOSUB LABEL is used in much the same way as a standard GOSUB command (it must be terminated with a **return** command).
DO control is used then the commands following the **if** command line within the **structure start and stop** will be executed.
REENT is used when the last **enter** command is to be redone.
THEN is used to execute the next line if the true/false test is true, otherwise the line will be skipped.

TAS EDITOR EXAMPLE:

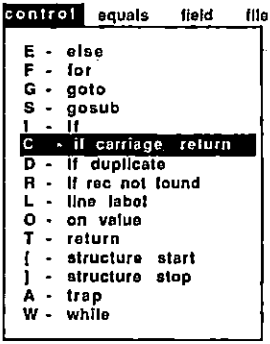
```
if: TEST = (TEST1) GOTO LABEL

if: TEST <= (10) GOSUB LABEL

if: TEST <> (TEST1 + 10) THEN
  (execute this next line)

if: TEST <> (TEST1 + 10) DO
{
(execute lines between braces)
}
```

if carriage return



The **if carriage return** command is used specifically in the source code editor to notify the program if the last entry was a carraige return or not.

PARTS OF THE COMMAND:

what to do if cr entered - If the carriage return is the last keyboard entry before this command, then the program will: G - goto a line label, S - gosub to a line label, or D - do the following structure.

enter line label - If goto or gosub is chosen, then this is the label of the line label where program control will be transferred.

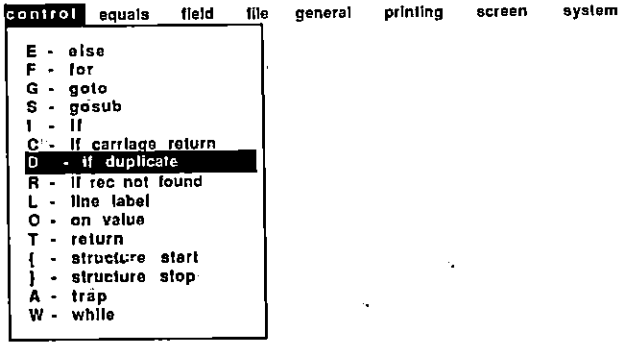
TAS EDITOR EXAMPLE:

```
if cr was entered: GOTO LABEL

if cr was entered: GOSUB LABEL

if cr was entered: DO
{
(execute lines between braces)
}
```

if duplicate



The **if duplicate** command is used to make sure a field value has not been duplicated in a previously entered record. This would generally be used to check a key field to prevent duplicate keys.

PARTS OF THE COMMAND:

fieldname - This is the field to be checked for a duplicate record. It may be any legal field name in the format *fieldname[ss].* It can not be an array field. The field must be designated a key field in the data dictionary. This is required.

what to do if true (GSD) - This is the action taken if there is a duplicate record. G - goto a line label, same effect as the **goto** command. S - gosub to a line label, same effect as the **gosub** command. D - do, this is the same as the do option for the **if** command. The program will execute the lines within the braces that immediately follow the **if duplicate** command line. This is required.

line label - In the case of the action being G (goto) or S (gosub), this is where program control will be transferred. It is not used with the D (do) option. A line label can be up to 10 character

long. It can only contain letters and numbers. Quote marks, math and other symbols are not allowed. This is required.

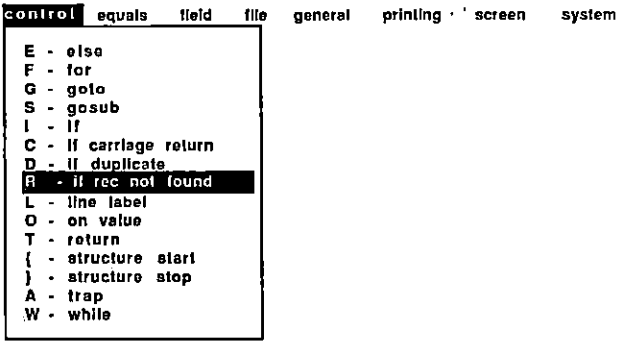
TAS EDITOR EXAMPLE:

```
if duplicate record: KEYFIELD GOTO LABEL

if duplicate record: KEYFIELD GOSUB LABEL

if duplicate record: KEYFIELD DO
{
(execute lines between braces)
}
```

if rec not found



The **if rec not found** command is used to check if a record has been successfully found and is now active (data is in the record buffer and the record number is not 0).

PARTS OF THE COMMAND:

filename - This is the name of the file to be checked. It is required.

what to do if not active (GSD) - This is the action taken if the record is not active (record number is 0). G - goto a line label, same effect as the **goto** command. S - gosub to a line label, same effect as the **gosub** command. D - do, this is the same as the do option for the **if** command. The program will execute the lines within the braces that immediately follow the **if rec not found** command line. This is required.

line label - In the case of the action being G (goto) or S (gosub), this is where program control will be transferred. It is not used with the D (do) option. A line label can be up to 10 character long. It can only contain letters and numbers. Quote marks, math and other symbols are not allowed. This is required.

TAS EDITOR EXAMPLES:

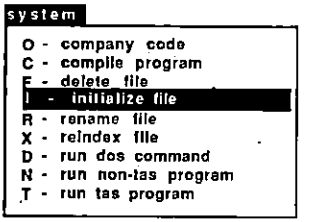
```
if record not found: TEST GOTO LABEL

if record not found: TEST GOSUB LABEL

if record not found: TEST DO
{
(execute lines between braces)
}
```

initialize file

control equals field file general printing screen system


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **initialize file** command is used to initialize a TAS file. This works the same as chosing number 4, "Initialize Data File" from the main TAS menu. If records already exist in the file, the command will ask if you want to initalize anyway. The program will return to the next line after the initialization is complete.

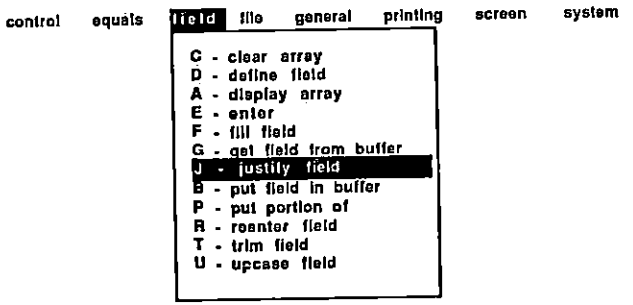
PARTS OF THE COMMAND:

file name - This is the name of the file to be initialized.

TAS EDITOR EXAMPLE:

initialize file: MYDATA

justify field



The **justify field** command is used to justify (align) the characters within a field to the left, right or center. If the number of non-space characters within the field is the same as the maximum field size this command will have no effect on the field.

PARTS OF THE COMMAND:

field to justify - This is the field to be justified. It is a legal alphanumeric field in the format of *fieldname[ss]*. This is required.

justify (LRC) - This tells whether to justifiy the characters to the left (L), to the right (R), or to the center (C) within the field boundries. This is required.

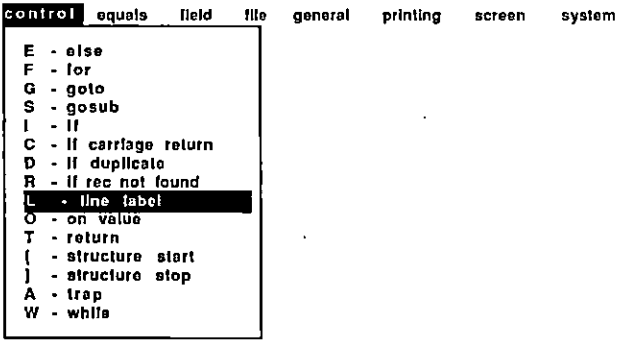
TAS EDITOR EXAMPLE:

justify: TEST LEFT

justify: TEST RIGHT

justify: TEST CENTER

line label



The **line label** command is used to create a destination for a **goto** command or a **gosub** command. Line labels can contain up to 10 alphanumeric characters. Characters can only be letters or numbers. Quote marks, math signs and symbols are not allowed.

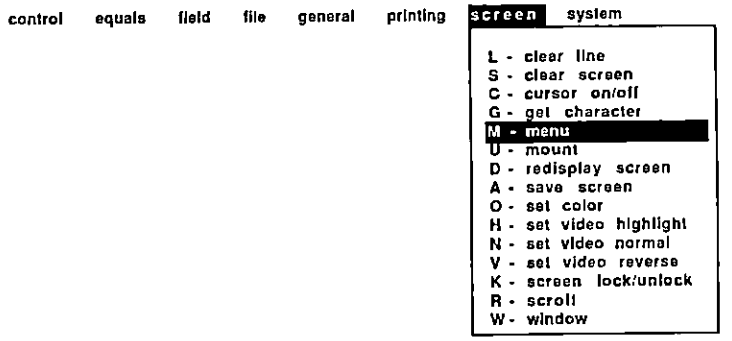
PARTS OF THE COMMAND:

enter the line label - This is where program will be trasferred to in a goto or gosub routine.

TAS EDITOR EXAMPLE:

START

menu



The **menu** command is similar to the **window** command and is used to create a moving bar type menu. The menu can be coded into the program or created "on the fly" at run time. The choices on the menu are stored in an alpha array field which may be defined in the program or may be a standard data file field. If the counter field is preset before executing the menu command, then the system will automatically highlight the appropriate item using reverse video.

A single character is specified as the "choice" for each item during setup. At runtime the item may be selected by typing the choice character or by moving the bar with the up/down arrow keys and hitting return. After a choice is made, the command will clear the screen and automatically return to the previous screen. The array number of the item choosen will be returned to the counter variable (i.e. if you chose item #7, then the number 7 is returned to the counter variable).

This was done so that if you could follow the **menu** command immediately with an **on** command and no further transliteration need be done. However, if you were pulling a character out of a group, then you could use the same counter field in an **equals portion of** command.

Note: Each element of the menu array field will be one choice in the menu. The item should be set up in the following manner: (' Name xxyyn'). It looks better if the first character in the item is a space, since this will give a little visual distance from the box around the menu. Then spaces would be placed after the name until you have a string that is the same length as the value you set for the maximum width of choice.

The proper convention is to capitalize the choice character. The next two characters are the column loction from 01 to 80 (box limits are 80 characters wide, 24 lines high). Next are the two row characters from 01 to 24. The final item is the choice character. You should also note that the menu does not have to be absolutely vertical or horizontal and may be made up of columns or rows as well as single line fields. The limit to the number of choices within one menu is 255.

PARTS OF THE COMMAND:

lower left column value - This is a numeric constant or numeric field using the standard format of *fieldname[ss].* Limits 1 - 80

lower left row value - This is a numeric constant or numeric field using the standard format of *fieldname[ss].* Limits 1 - 24

height - This is a numeric constant or numeric field using the standard format of *fieldname[ss].* The limit is 22 or the lower left row value minus two.

width - This is a numeric constant or numeric field using the standard format of *fieldname[ss].* The limit is 79 or the lower left column value subtracted from 80.

menu array field - This is a numeric constant or numeric field using the standard format of *fieldname[ss].* The name of the array field where

the menu choice names are stored. This field must be defined to have at least as many array elements as you have menu choices.

choice field - This is a numeric constant field using the standard format of *fieldname[ss]*. The name of the variable in which to store the user's choice. This can be used with an **if** or **on** command to direct action of the program.

maximum number of choices - This is a numeric constant field using the standard format of *fieldname[ss]*. The limit is 255.

maximum width of choice - This is a numeric constant field using the standard format of *fieldname[ss]*. The number of characters in the widest choice you will be using.

where to goto if ESC is pressed - This is any valid line label. If the user presses the escape key as the menu response, the program will go to this label.

where to gosub if HELP is pressed - This is any valid line label. If the user presses the help key as the menu response, the program will go to this label. Don't forget to return from the subroutine with a RETURN command.

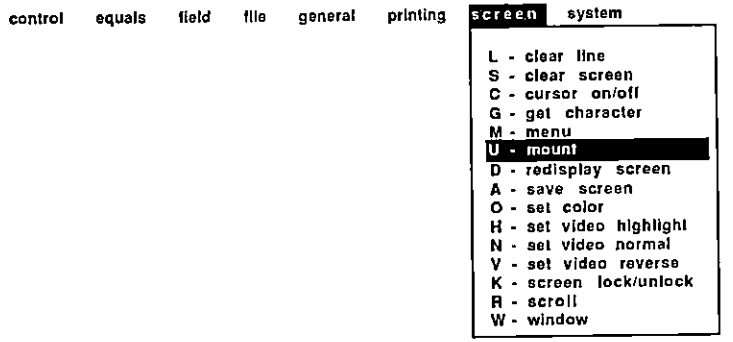
menu title - This is any valid alpha field or constant. It will be displayed within the upper line of the box surrounding the menu lines.

TAS EDITOR EXAMPLE:

```
menu: column: 10 row: 15 height: 7 width: 20
menu field: ARRAYFLD receiving field: CNTR
max choices: 6 max width: 19 on ESC goto: QUIT
if help gosub: WHAT title: ' Multiple Menu '

menu: column: COLFIELD row: ROWFIELD
height: HGTFIELD width: WDTFIELD menu field:
ARRAYFLD receiving field: CNTR max choices:
MCFIELD max width: MWFIELD on ESC goto:
QUIT if help gosub: WHAT title: TITLEFIELD
```

mount



The **mount** command is used to display a selected screen on the terminal if the S (screen) type is selected. If the R (report) type is selected it will activate a report format. There may be multiple formats of either type in a single program, however, only one format may be mounted at a time.

PARTS OF THE COMMAND:

format name - This is the name you have assigned the format in the program. This can be up to eight alphanumeric characters. It is the name you assigned when you created the screen. It is required.

which type (S/R) - This tells what type of format. Screen (S) or (R) report. It is required.

where to print (ASPD) - If it is a report format, then this determines where to send the report. A - ask at runtime, S - print to screen or crt, P - print on printer, D - send it to the disk. This is required.

file path/name - If it is a report format and D is chosen this is the disk file name. The name should be a standard MS-DOS file name and may include a path name.

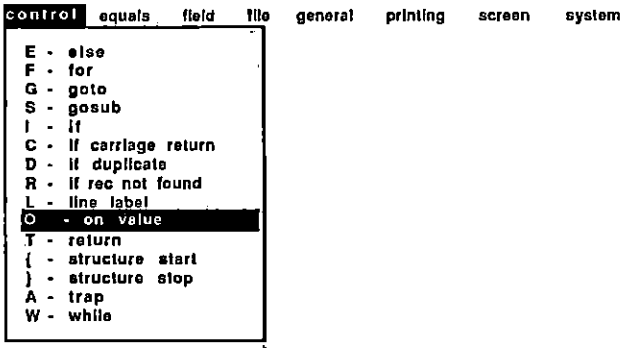
TAS EDITOR EXAMPLE:

load format: TEST type: S

load format: TEST type: R print on: A

load format: TEST type: R print on: D REPORT

on value



The **on value** command will transfer control to different line labels depending on the value of a numeric field. If the value of the field is 1 then the program will transfer control to the first line label, if 2 to the second, etc. If the value is greater than the number of line labels, then control is transfered to the error label.

PARTS OF THE COMMAND:

on field - This is a legal numeric field in the format of *fieldname[ss]*. This is required.

what to do (GS) - This determines the type of transfer.
G - goto (same as the standard **goto** command.
S - gosub (same as the standard **gosub** command). This is required.

label1;label2;...;labelx - The line labels to transfer control to depending on the value of the ON FIELD. If the value is 1 then the program will transfer control to LABEL1, if 2 then LABEL2 etc. This is required.

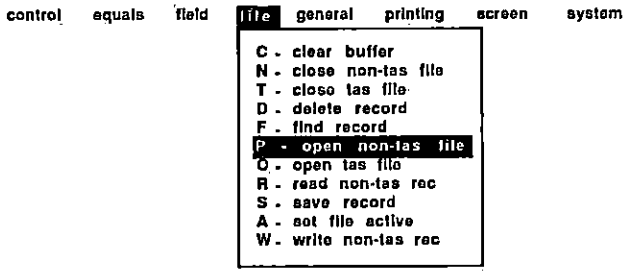
error label - If the value of ON FIELD is greater than the number of labels (excluding the error

label), then control will be transfered to the error label.

TAS EDITOR EXAMPLE:

on: TEST GOTO FIRST;SECOND;THIRD;ETC. if error goto: LABEL

open non-tas file



The **open non-tas file** command is used to open a non-TAS file before any records are read from or written to that file. Three types of non-TAS ASCII files are supported. They are quote/comma delimited files (such as Wordstar Mailmerge files), fixed length files (used by many databases to store data), and standard text files (such as those created by Wordstar in its non-document mode).

This command works with the **read non-tas rec** command and the **get field from buffer** command to allow data to be imported from other programs into a TAS program.

PARTS OF THE COMMAND:

type of file (FTD) - This is the type of record in the file to be opened. F - fixed length ASCII records, T - text records (each record must be terminated with a CR/LF), and D - ASCII data records where each field is delimited by commas (and quotes where necessary) and each record is terminated with a CR/LF. This is required.

filename - This is the file to be opened. It may be a standard alphanumeric field in the format of *fieldname[ss]* or an alphanumeric constant in the

form of a legal MS-DOS path and file name. This is required.

buffer start - The buffer for the record to be read must be a defined field. If the record is longer than 255 characters, then multiple fields must be defined and should be in contiguous order. The first field in the group (which may be the only field) is considered the BUFFER-START and the name of that field would be put at this location. This is required.

record size - This must be a numeric constant of no more than 5 characters. The maximum size of the record to be read is 65535 characters. This is required.

NOTE: Make sure you have defined a large enough field (or group of fields) to handle the largest record possible. If you read a record into a buffer that is smaller than the original record, the results will be unpredictable.

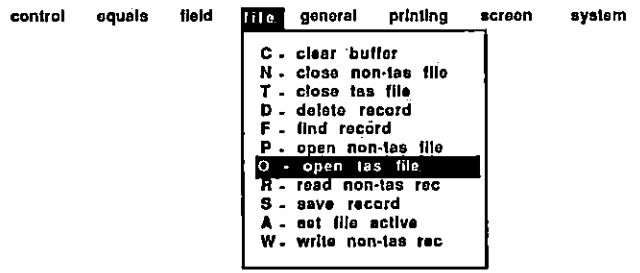
TAS EDITOR EXAMPLE:

open non-tas file type: F filename: TEST.FIX
buffer start: FIELD1 record size: 120

open non-tas file type: T filename: TEST.TXT
buffer start: FIELD1 record size: 120

open non-tas file type: D filename: TEST.DTA
buffer start: FIELD1 record size: 120

open tas file



The **open tas file** command is used to open a TAS file before any records may be read from or written to that file.

PARTS OF THE COMMAND:

filename - This is the name of the file to be opened. A record for this file must be in the locator file and the actual key and master files must be at the locations specified. It must be the actual filename and not a variable. This is required.

company code - This sets or removes the company code extention. It may be any legal alphanumeric field in the format of *fieldname[ss]* or a two character alphanumeric constant enclosed with quotes. It can also be C, which sets the value to company code specified at the main menu (or defined with the **company code** command). It may also be S, which would be no extension. The default is C.

lock type - This is used to lock a file in a multiuser application. The options are 0 - no locks, 1 - lock a record when read, or 2 - do not allow any additions or changes to the file (this

would effectively lock the file while still allowing search access). The default is 0.

NOTE: If you think the program may be used on a multi-user system then you should specify locking type 1. It will run properly when used on a single-user system. If, however, it is being run multi-user then the locking requirements will be taken care of.

error label - This is where control will be transferred if an error occurs when opening the file. This should be a message followed by an exit from the program since any further attempt at file access will give the error that the file is not active.

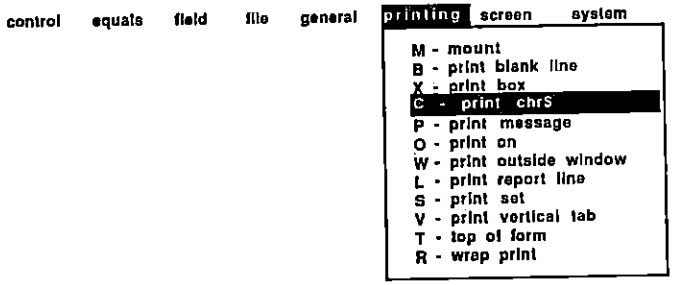
TAS EDITOR EXAMPLE:

open: TEST NO LOCKS on open error: LABEL

open: TEST co code: AA RECORD LOCKS on error: LABEL

open: TEST LOCK FILE on open error: LABEL

print chr$



The **print chr$** command is used to send a string of control codes to the printer. These are characters previously set up with the installation program (in printer control code section). They are generally used for printing enhanced formats (such as bold, condensed, etc.). To set up the character strings please see Appendix VI, Installing and Customizing TAS-Plus.

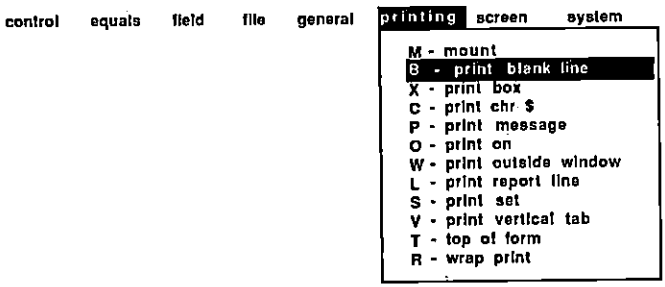
PARTS OF THE COMMAND:

character string number - This is number preassigned to the character string during installation. It is a numeric constant from 1 to 20.

TAS EDITOR EXAMPLE:

print chr string: 3

print blank lines



The **print blank lines** command is used to print a number of blank lines to the active device. This could be the screen, the printer, or the disk.

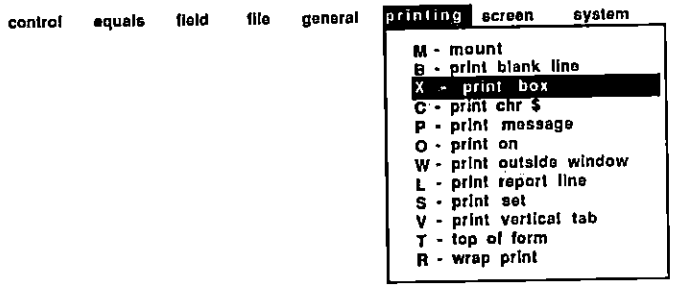
PARTS OF THE COMMAND:

number of lines - This is the number of lines to be printed. It is a numeric constant between 1 and 255. This is required.

TAS EDITOR EXAMPLE:

print blank lines: 10

print box



The **print box** command is used to print a box on the screen or to clear an existing box from the screen. This command can be used to decorate your input screens or to do simple graphics representations of data (bar graph).

PARTS OF THE COMMAND:

column - This is the column number of the lower left corner of the box. It is a numeric field or constant between 1 and 80. This is required.

row - This is the row number of the lower left corner of the box. It is a numeric field or constant between 1 and 24. This is required.

height - This is the number of rows tall the box will be. It is a numeric field or constant between 1 and 24. This is required.

width - This is the number of columns wide the box will be. It is a numeric field or constant between 1 and 80. This is required.

clear box (YN) - Y - will clear the box on the screen. N - will display the box on the screen. The default is N.

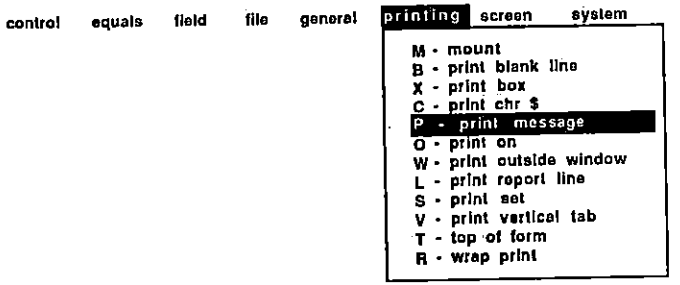
TAS EDITOR EXAMPLE

print box: column: 10 row: 20 height: 18 width: 40

print box: column: COL row: LIN height: HITE width: WID

print box: column: 10 row: LIN height: 4 width: WID CLEAR

print message



The **print message** command is used to print a message (or field) on the screen . It is commonly used to display help or error messages and to enter display fields on the screen.

Note: You can put control character directly into the line to be printed. This is done be putting a % directly in front of a 1 to 3 character numeric value. For example, a carriage return code would be **%13**.

You may also change the color by putting the @ symbol in front of the character R (for reverse video), H (for highlighted video), or N (for normal video). For example, @R in the message would turn on the reverse video before the balance of the message was sent. You may also specify a certain color value by using @C and then the color value number from 1 through 255 (for example, @C7 would set the color to normal on a monochrome system).

As in a normal message these options should be separated from the balance of the message by semicolons (;).

PARTS OF THE COMMAND:

message - A message may be an alphanumeric

constant surrounded by quotes, such as 'Press any key to continue', or a field name in the format of *fieldname[ss]*, or a combination of both as in

'total sales';totsales

If more than one message item is to be used in the same **prt message** command, then they must be separated with a semicolon (;). The maximum message length is 255 characters per command line.

column value - This is the column where the message will begin. It may be a numeric constant or a numeric field in the format of *fieldname[ss]*. This is only effective if the message is going to the screen. The default is to continue from last cursor position.

row value - This is the row where the message will begin. It may be a numeric constant or a numeric field in the format of *fieldname[ss]*. This is only effective if the message is going to the screen. The default is to continue from last cursor position.

wait for chr entry (YN) - Y - the program will wait until the operator has pressed any key before continuing. N - program will procede without waiting. This is only effective if the message is going to the screen. The default is N.

do not do carriage return (YN) - Y - a CR/LF will not be sent after the message is printed. N - a CR/LF will be sent after the message is printed. This would allow a second message to be printed on the same line. The default is N (a CR/LF is sent).

force output where (SPD) - This will force the message to go to the screen (S), printer (P) or default (D) device. For example, if you were printing a report and wanted to send an error

message to the screen, then this would be S. The default is D.

entry field - If you name a variable, TAS will let you enter its value on the screen after the message appears. This allows you to enter data without mounting a separate screen.

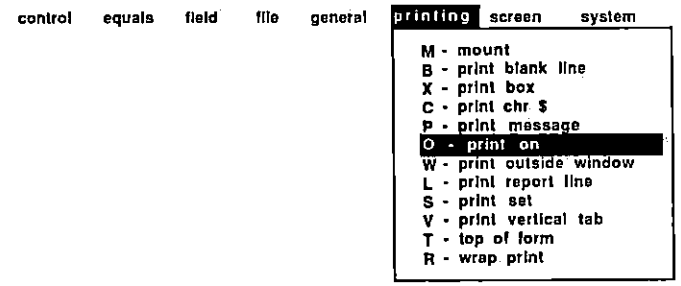
TAS EDITOR EXAMPLE:

prt message: 'This is a test' column: 1 row 24

prt message: The amount due is $'; AMOUNT NO C/R force to: PRINTER

prt message: 'Please enter the amount due:' column: 1 row: 10 force to: SCREEN entry field: AMOUNT

print on



The **print on** command is used to set the default device for printing and messages without having to mount a report format.

PARTS OF THE COMMAND:

print on which device (ASPD) - This is where to send the printing. A - ask at run time, S - send to screen, P - send to printer, D - send to disk. This is required.

filename - If the above choice is D, then you may also enter a filename either as an alphanumeric constant or a field in the format of *fieldname[ss]*.

TAS EDITOR EXAMPLE:

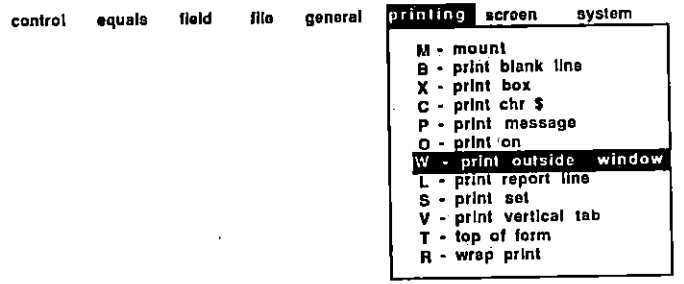
print on: ASK AT RUN TIME

print on: CRT SCREEN

print on: PRINTER

print on: DISK FILE filename: TESTPRT

print outside window



The **print outside window** command lets you print a message on screen that is outside of the window. This command is typically used with the **print message** command.

PARTS OF THE COMMAND:

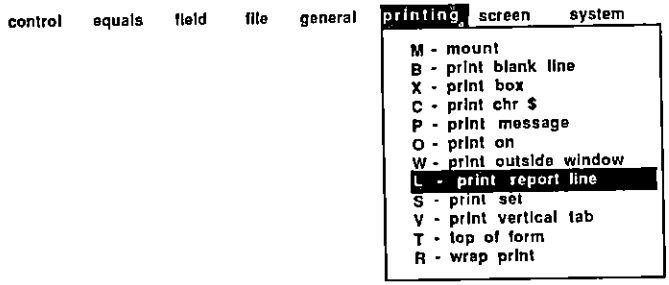
force ? (YN) - Y - allows printing messages on screen outside of the currently active window. N - restricts messages to being printed inside the currently active window.

TAS EDITOR EXAMPLE:

force printing outside window: Y

force printing outside window: N

print report line



The **print report line** command is used to print a specified line from a report format. The report format must be mounted with the **mount** command before this command is used.

PARTS OF THE COMMAND:

report format line - This is the format line number to be printed. Each line in the report format is a separate format line and has its own number from 1 to 66. It may be a numeric constant or a numeric field in the format of *fieldname[ss]*. This is required.

column value - This is the column where the printing will begin. It may be a numeric constant or a numeric field in the format of *fieldname[ss]*. This is only effective if the printing is going to the screen. The default is to continue from last cursor position.

row value - This is the row where the printing will begin. It may be either a numeric constant or a numeric field in the format of *fieldname[ss]*. This is only effective if the printing is going to the screen. The default is to continue from last cursor position.

wait (YN) - Y - the program will wait until the operator has pressed any key before continuing. N - The program will not wait for operator. This is only effective if the printing is going to the screen. The default is N.

no CR (YN) - Y - a CR/LF will not be sent after the line is printed and the cursor will remain on the line of the printing. N - a CR/LF will be sent. This would allow a second message to be printed on the same line. The default is N (a CR/LF is sent).

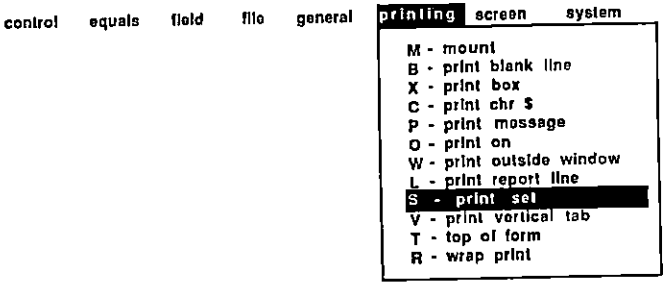
force output where (SPD) - This will force the message to go to the screen (S), printer (P) or default (D) device. For example, if you were printing a report and wanted to send an error message to the screen, then this would be S. The default is D.

TAS EDITOR EXAMPLE:

prt report line: 1 column: 1 row: 1 force to: SCREEN

prt report line: RPTFIELD column: COLFIELD row: ROWFIELD WAIT NO C/R

print set



The **print set** command is used to change the default print settings in the current program. The defaults will be restored when a new program is run. They are 132 columns print width, 60 lines maximum print length, and 66 lines maximum paper length. If the screen is chosen as the print device then the defaults change to 78 columns print width, 24 lines maximum print and paper length.

Note: The **print set** command must be placed after the **screen mount** command for it to have an effect on a report format. Otherwise the screen mount will reset the default number of lines.

PARTS OF THE COMMAND:

max width - This is the maximum width of the paper or print device. It must be a numeric constant. This is required.

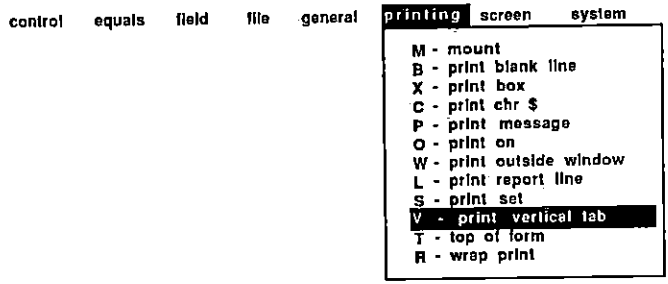
max lines - This is the maximum number of lines on the paper or print device. It must be a numeric constant. This is required.

max print lines - This is the maximum number of printable lines on the paper or print device. It must be a numeric constant. This is required.

TAS EDITOR EXAMPLE:

set format controls: width: 255 tot lines: 88 print lines: 80

print vertical tab



The **print vertical tab** command is used to do a vertical tab and to advance to a specified line number. If the line number is less than the current location or if it is beyond the end of the form, it will be ignored .

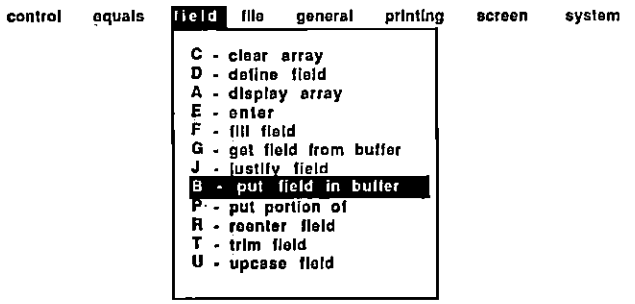
PARTS OF THE COMMAND:

line number - This is the line number to be advanced to. It must be a numeric constant. This is required.

TAS EDITOR EXAMPLE:

vertical tab: 30

put field in buffer



The **put field in buffer** command is used to place a field in the next position in a buffer when creating quote/comma delimited records (such as a mailmerge data file). If the field is an alphanumeric and contains a comma, then it will be enclosed by two quotes ("field, value"). In all cases, a field will be separated from the previous field by a comma. Before the record is written it will be terminated with a CR/LF. If the field to be added is a numeric or date it will be converted to alpha-numeric format before it is added to the record. Also, any leading spaces in a numeric field or trailing spaces in an alphanumeric field will be deleted.

Note: To make sure your record buffer is clear, you should begin your series of **put field in buffer** commands with a blank fieldname. This command works with the **write non-tas rec** command to send tas data out to non-tas programs.

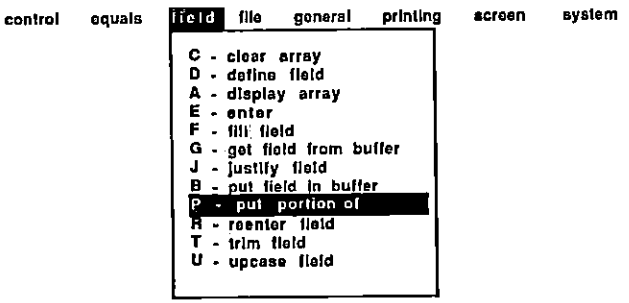
PARTS OF THE COMMAND:

field to be added to record - This is any legal field in the format of *fieldname[ss]*. Each field will be added to the record in the order that this command is executed. If it is left blank the field will be cleared.

TAS EDITOR EXAMPLE:

place field: CLEAR FILE BUFFER

put portion of



The **put portion of** command is used to insert a string of characters into a larger string of characters. It is similar to the **equals portion of** command except that instead of taking a part of a larger field, this command will insert a smaller field into a larger.

Note: When you use this command to insert a string of characters, they will replace the existing characters that occupied the same positions in the receiving field.

PARTS OF THE COMMAND:

receiving field - This is the field that will receive the characters. It must be an alphanumeric field in the format *fieldname[ss].*

starting character - This is the position in the receiving field where the new character string will start. This can be a numeric constant or a numeric field in the format *fieldname[ss].*

number of characters - This is the length of the character string that is being inserted into the receiving field. This can be a numeric constant or a numeric field in the format *fieldname[ss].*

field to be inserted - This is the field that contains the character string to be inserted into the receiving field. This can be an alpha constant or a legal field in the format *fieldname[ss]*.

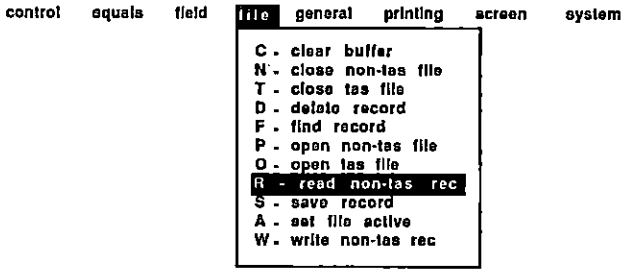
Note: If the field to be inserted is anything other than an alpha constant or alpha field (such as date, time, numeric, etc.), then the BCD equivalent and not the alpha representation will be inserted. This is important since BCD numbers contain fewer characters than their alpha representations.

TAS EDITOR EXAMPLE:

put in field: RECFIELD start chr: 1 num chrs: 5 insert filed: 'ABCDE'

put in field: RECFIELD start chr: STARTFIELD num chrs: LTHFIELD insert filed: INSFIELD

read non-tas rec



The **read non-tas rec** command is used to read a record from a non-tas file into the buffer specified in the **open non-tas file** command. The command always reads sequentially starting with record number one.

Note: See the **get field from buffer** command and **open non-tas file** command.

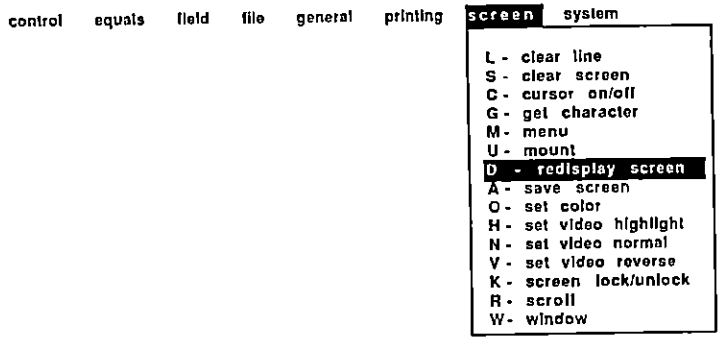
PARTS OF THE COMMAND:

on error goto label - This is where program control will be transferred if a file error occurs during the read.

TAS EDITOR EXAMPLE:

read a record in non-tas file on error goto: LABEL

redisplay screen



The **redisplay screen** command is used to redisplay a screen that was previously saved to a buffer with the **save screen** command. See **window** command and **save screen** command.

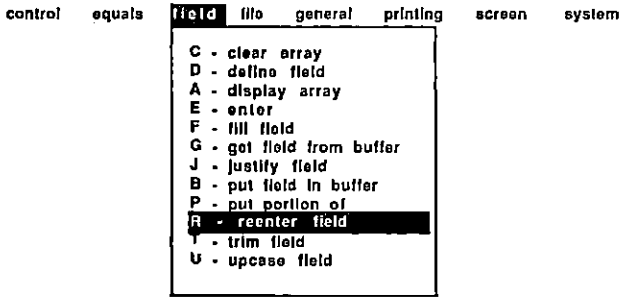
PARTS OF THE COMMAND:

Buffer number - This is which buffer the screen was saved in. It must be a numeric constant between 1 and 10.

TAS EDITOR EXAMPLE:

redisplay screen previously saved - number: 1

reenter field



The **reenter** command is used to redo the last enter command executed.

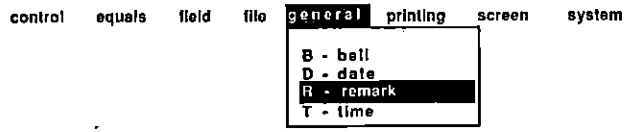
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

re-do last enter command

remark



The **remark** command is used to place a remark line in your program. This line is not active and will not be compiled. A single remark may be up to 144 characters long.

PARTS OF THE COMMAND:

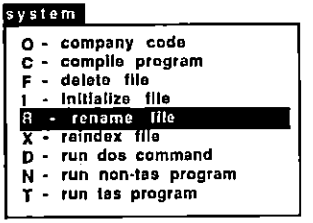
rem: - This is the text of the remark.

TAS EDITOR EXAMPLE:

rem: THIS IS A SAMPLE REM STATEMENT

rename file

control equals field file general printing screen **system**


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **rename file** command is used to rename a file on the disk. You may use any legal MS-DOS path names and wildcards. This would be the equivalent of using the DOS REN command.

PARTS OF THE COMMAND:

current name - This is the name of the file to be changed. It may be an alpha constant or alpha field in the format of *fieldname[ss]*. It may include standard MS-DOS wildcard notations (such as * or ?). It may also be preceeded by a path name.

new name - This is the new name of the file. It may be an alpha constant or alpha field in the format of *fieldname[ss]*. It may include standard MS-DOS wildcard notations (such as * or ?). It may also be preceeded by a path name.

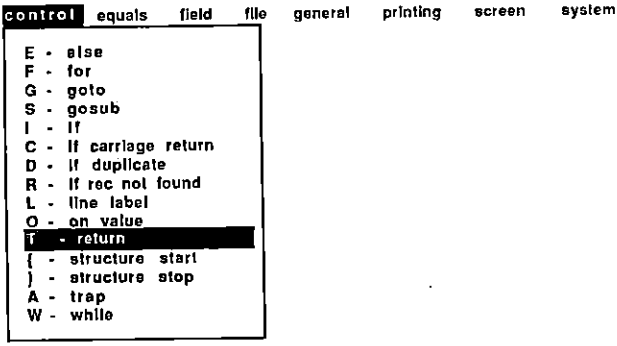
TAS EDITOR EXAMPLE:

rename file from: 'TEST.EDT' to: 'TEST.HLD'

rename file from: '*.K' to: '*.KCC'

rename file from: OLDFILE to: NEWFILE

return



The **return** command is used to return from a subroutine that was reached with a **gosub** command. If an attempt is made to return without previously executing a **gosub**, an error will be displayed and the program will exit to the main menu.

PARTS OF THE COMMAND:

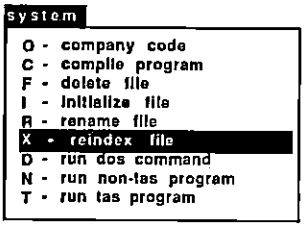
none

TAS EDITOR EXAMPLE:

return

reindex file

control equals field file general printing screen system


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **reindex file** command is used to reindex the TAS file for the current number of records. The program will return to the next line after the reindexing is complete.

Note: This is the same as chosing number 3 "Re Index Data File" from the main TAS menu (in the Utilities section).

PARTS OF THE COMMAND:

File name - This is the name of the TAS file to be reindexed. It may be an alpha constant or alpha field in the format of *fieldname[ss].*

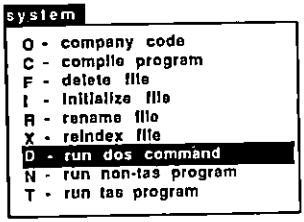
TAS EDITOR EXAMPLE:

reindex file: 'TASFILE'

reindex file: TESTFILE

run dos command

control equals field file general printing screen system


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **run dos command** command is used to execute a standard MS-DOS command (such as DIR) from within TAS. After the MS-DOS command is finished (and you have pressed any key to continue), you will be returned to the next line of the TAS program.

PARTS OF THE COMMAND:

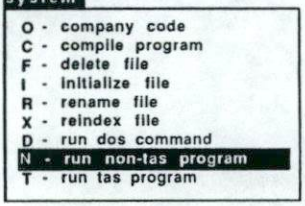
command line - This is the MS-DOS command to be executed. It may be an alpha constant or alpha field in the format of *fieldname[ss]*.

TAS EDITOR EXAMPLE:

execute dos cmd: 'DIR C:'

run non-tas program

control equals field file general printing screen **system**


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **run non-tas program** command is used to execute a .COM or .EXE program. When the non-tas program is finished, TAS will resume processing with the next line in the TAS program. The data existing before the non-tas progarm was run will still be intact. If there is an error in running the non-tas program (i.e. "file not found" or "program to big" errors), then TAS will execute the next line of the original program as if the non-tas program had executed properly.

PARTS OF THE COMMAND:

path and program name - This is an alphanumeric constant or any legal field using the standard format of *fieldname[ss]*. It may also be preceeded by a path name.

command trailer - This is an alphanumeric constant or any legal field using the standard format of *fieldname[ss]*.

TAS EDITOR EXAMPLE:

execute program: TESTPROG.COM

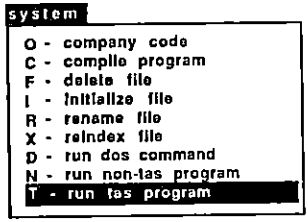
execute program: B:TEST.COM

execute program: TESTPROG.EXE TESTFILE

execute program: \TESTDIR\TESTPROG.COM

run tas program

control equals field file general printing screen system


O - company code
C - compile program
F - delete file
I - initialize file
R - rename file
X - reindex file
D - run dos command
N - run non-tas program
T - run tas program

The **run tas program** command is used to execute a TAS program.

PARTS OF THE COMMAND:

path and program name - This is the name of the TAS program to be executed. It is an alphanumeric constant or any alphanumeric field in the format *fieldname[ss].* the extension is automatically set to .RUN. To return to the language menu at the end of a program, you should place a **run** command (with no program name) at the end of your program.

Note: The following are special program names that are used to do specific predefined functions in TAS-Plus. Each one is two alpha characters surrounded by quote marks.

'MM' - will run program designated as main menu.
'RI' - will run reindex program.
'RS' - will run restructure program.
'EX' - will exit to DOS.
'RT' - will ask for TAS program to run.

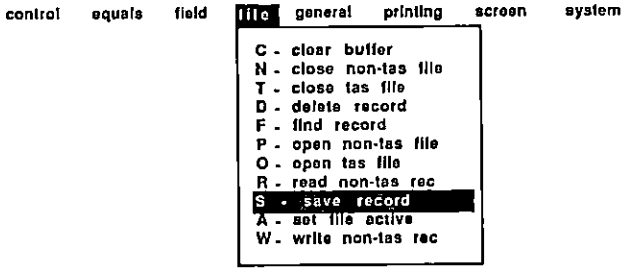
TAS EDITOR EXAMPLE:

run tas program: "MM"

run tas program: "C:\MYDIR\MYPROG"

run tas program: PROG.FIELD

save record



The **save record** command is used to save a record in a TAS file. The program will automatically update the keys in the record if there are any. If this is a new record it will use the space from a deleted record (if there is any) before it adds to the file size.

PARTS OF THE COMMAND:

filename - This is the filename the record is to be saved to. It must be the actual file name and not a variable. Qoute marks are not required. This is required.

confirm (YN) - Y - will ask whether to do the save. N - will not ask. The default is Y.

clear buff (YN) - This will clear the buffer after the save. If the buffer is not cleared, then the record number is not cleared. If this occurs and the record is saved again then the program will just re-save the same record and will not create a new one. If you want to use the same data in another record then don't clear the buffer (N) but immediately after the **save record** command do a **clear buffer** command (with R option).

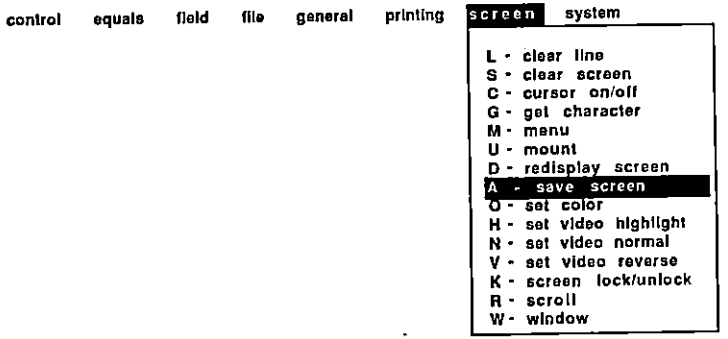
goto label - If the confirm option is Y, this is where program control will be transferred if a record is not saved at runtime.

TAS EDITOR EXAMPLE:

save record: TEST confirm: Y clear buffer: Y goto label: LABEL

save record: TEST confirm: N clear buffer: Y

save screen



The **save screen** command is used to save the current screen in a buffer for redisplaying later. Screens may be saved in up to 10 different buffers at a time. Once a buffer is used it may be reused later. When you reuse a buffer with a new screen, the original screen will be lost. See **redisplay** command and **window** command.

Note: By using this command, the window command, and the redisplay screen command, you are able to present the effect of overlapping windows.

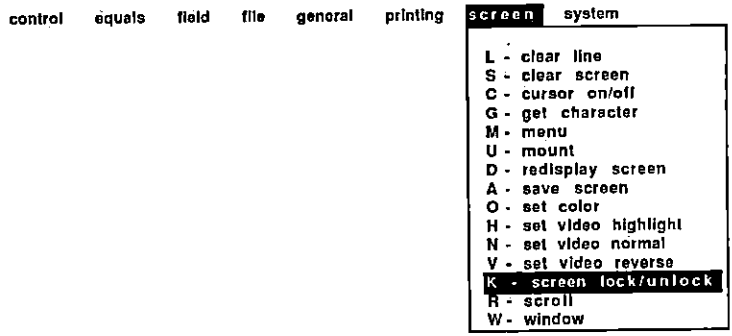
PARTS OF THE COMMAND:

buffer number - This is the specific buffer to which the screen will be saved. It is a numeric constant between 1 and 10.

TAS EDITOR EXAMPLE:

save screen for redisplay - number: 1

screen lock/unlock



The **screen lock/unlock** command is used to lock, unlock or redisplay the information on the screen. If the screen is locked, then no information appearing on the screen will be updated, even if a new record is read and data is being updated internally. When the screen is unlocked, all fields will be redisplayed. All the information will be redisplayed if the R option is chosen.

PARTS OF THE COMMAND:

what to do (LUR) - L - locks the screen, U - unlocks the screen, R - redisplays information on the screen. The redisplay option is usually used in conjuction with the **clear array** command.

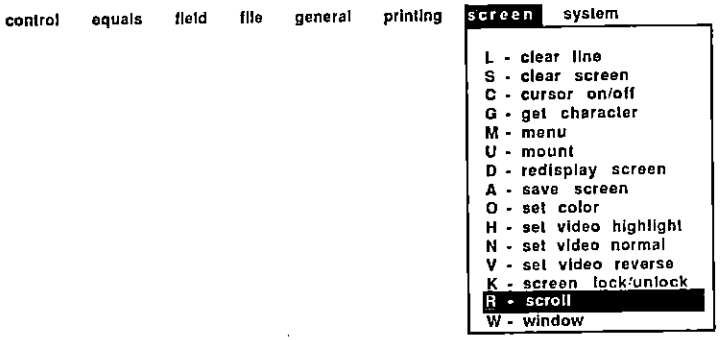
TAS EDITOR EXAMPLE:

screen: LOCK

screen: UNLOCK

screen: REDISPLAY

scroll



The **scroll** command is used to move the information on the screen up, down, left or right within a certain area. This command can be used if you have a lot of information that you don't want to keep in memory but do want to be able to move arround on the screen. The size of the block to be scrolled will limit how many lines or columns you can move and still stay on the screen.

PARTS OF THE COMMAND:

location column value - This is the column value for the lower left corner of the box to be scrolled. It may be a numeric constant or numeric field in the format *fieldname[ss]*.

location row value - This is the row value for the lower left corner of the box to be scrolled. It may be a numeric constant or numeric field in the format *fieldname[ss]*.

height value - This is the number of rows high of the box to be scrolled. It may be a numeric constant or numeric field in the format *fieldname[ss]*.

width value - This is the number of columns wide of the box to be scrolled. It may be a

numeric constant or numeric field in the format *fieldname[ss]*.

number of lines/columns - This is the number of rows/columns that are to be moved. It may be a numeric constant or numeric field in the format *fieldname[ss]*.

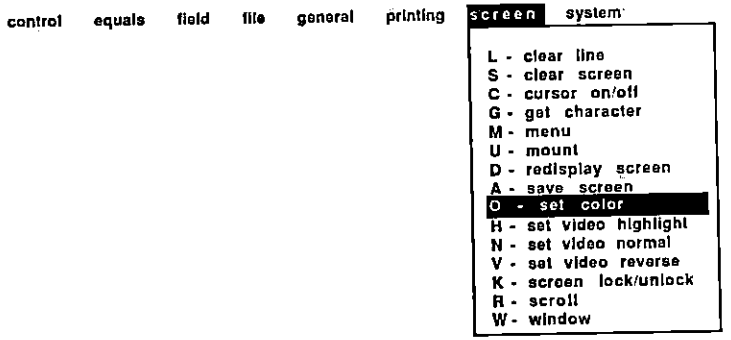
which way (UDLR) - U - allows scrolling up. D - allows scrolling down. L - scrolls left. R - scrolls right.

TAS EDITOR EXAMPLE:

scroll: column: COL row: ROW width: HITE height: 40 # lines: MOVE UP

scroll: column: 24 row: 1 width: 24 height: 80 # lines: 22 LEFT

set color



The **set color** command is used to set the normal, highlighted, and reversed video colors. This command allows you to set system default values in the installation program. Colors may also be changed during program processing by referencing a numeric or field value .

NOTE: For more information on color please see Appendix V, Adding color and highlights to TAS programs.

PARTS OF THE COMMAND:

normal color value - A numeric value between 1 and 128 or any legal field using standard format of *fieldname[ss]*.

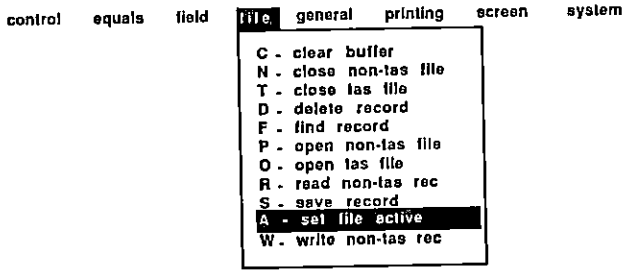
highlighted color value - A numeric value between 1 and 128 or any legal field using standard format of *fieldname[ss]*.

reverse color value - A numeric value between 1 and 128 or any legal field using standard format of *fieldname[ss]*.

TAS EDITOR EXAMPLE:

set color values: normal: 1 highlighted: 2 reverse: 3

set file active



The **set file active** command is used to specify which file will be searched by a **find record** command. It is specifically used when two or more files share the same schema (record layout).

TAS is set up to always open one file per schema. Normally this is no problem since each file usually has its own schema. It is, however, possible for files to share the same schema. In this instance it is necessary to tell the system which file is searched for a record.

The program defaults to the first file opened (that uses the buffer) as the active file. If you search for a record using a key field, then it will always search the first opened file. However, if you change files using the **set file active** command, it will search in which ever file you set active.

Note: You should always use the **open tas file** command to open the major file first. This will automatically set it as the active file. Also, the **set file active** command has no effect on where records are saved since the **save record** command asks you to specify a file.

PARTS OF THE COMMAND:

schema name - This is the schema name (record layout name) the file will be use as the buffer.

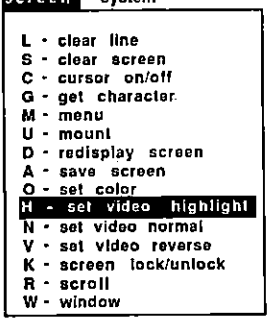
file name - This is the name of the file that will now be the active file.

TAS EDITOR EXAMPLE:

set file active: schema: BTAR file name: BTAP

set video highlight

control equals field file general printing **screen** system



The **set video highlight** command is used to set the terminal so that any further displays (such as using the **print message** command) will print using the highlighted video mode. Another way to turn on the highlight feature in a print message statement is to place a @H before the portion you want highlighted. @N returns it to normal.

Note: Different colors can be displayed on a color monitor by specify color control codes in the "highlight video" and "return to normal video" sections of TINSTALL program. See Appendix VI, Installing TAS-Plus.

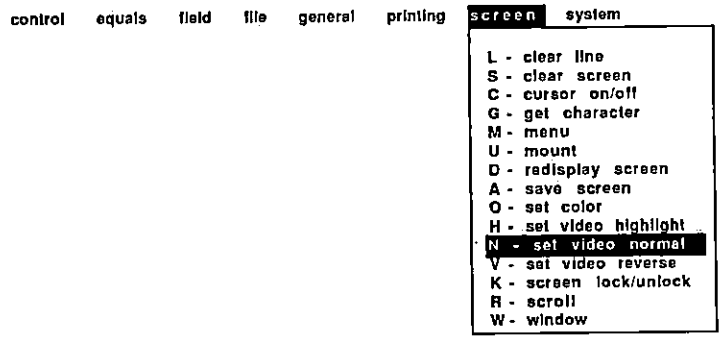
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

set display to highlighted mode

set video normal



The **set video normal** command is used to set the terminal so that any further displays (such as using the **print message** command) will print using the normal video mode. Another way to return to normal video in a print message statement is to place a @N at the point you want to switch into normal.

Note: See **set video highlight** command and **set video reverse** command.

PARTS OF THE COMMAND:

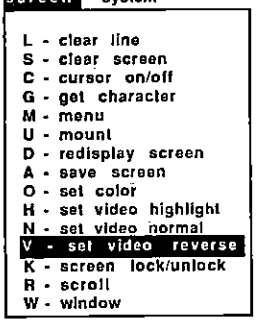
none

TAS EDITOR EXAMPLE:

set display to regular mode

set video reverse

control equals field file general printing **screen** system



The **set video reverse** command is used to set the terminal so that any further displays (such as using the **print message** command) will print using the reverse video mode. Another way to turn on the reverse feature in a print message statement is to place a @R before the portion you want highlighted. @N returns it to normal.

Note: Different colors can be displayed on a color monitor by specify color control codes in the "highlight video" and "return to normal video" sections of TINSTALL program. See Appendix VI, Installing TAS-Plus.

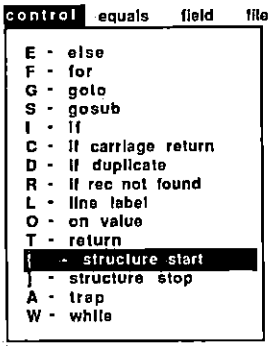
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

turn reverse video on

structure start



The **structure start** command is used to define the beginning of a set of commands that will be exectued as a group. This group of commands must end with a **structure stop** command.

The structure command is used with **for/next, if, else** and **while** commands to outline the procedure that will be executed if a true/false test is true. Structures may be nested 10 levels deep.

Note: For more information see section XII, Advanced TAS-Plus topics in the tutorial.

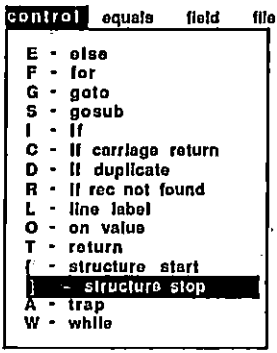
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

```
{
```

structure stop



The **structure stop** command is used to define the ending of a set of commands that will be exectued as a group. This group of commands must begin with a **structure start** command.

The structure command is used with **for/next, if, else** and **while** commands to outline the procedure that will be executed if a true/false test is true. Structures may be nested 10 levels deep.

Note: For more information see section XII, Advanced TAS-Plus topics in the tutorial.

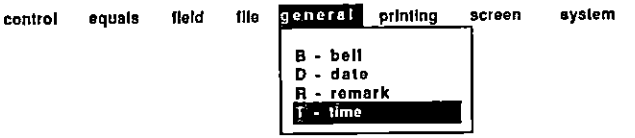
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

```
}
```

time



The **time** command is used to get the time from the system clock and put it in a data field. It is also used to set the system clock from a field.

PARTS OF THE COMMAND:

time field name: This a field of type T using the standard format of *fieldname[ss]*.

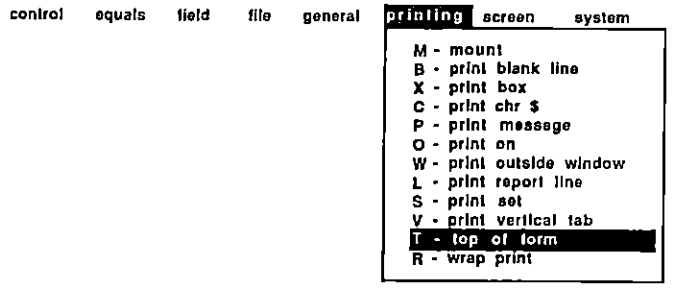
get or set: G - get system time. S - set system time.

TAS EDITOR EXAMPLE:

system time: field: TEST GET

system time: field: TEST SET

top of form



The **top of form** command is used to advance the paper to the top of the next form if the printer (or disk file) is active. It will also clear the screen and wait for a key to be pressed if the screen is active.

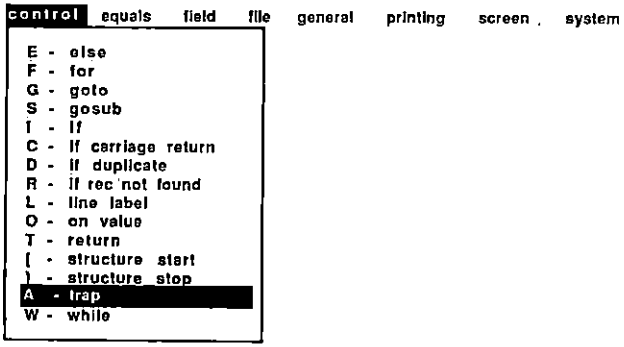
PARTS OF THE COMMAND:

none

TAS EDITOR EXAMPLE:

page to top of form

trap



The **trap** command is used to alert the program that a certain event has occurred and to tell the program what action to take. This event may be an input from the keyboard or an event generated by the program.

PARTS OF THE COMMAND:

type of trap - This is the event that will activate the command:
B - beginning/ending of file has occured.
C - clear program control code is entered.
D - delete program control code is entered.
F - file search (of any type) program control code is entered.
H - help program control code is entered
I - interrupt execution (ESC key is pressed while the program is running - but not at an **enter** command).
J - jump program control code is entered.
L - attempt to read a record that is locked has occurred.
N - next/previous is entered.
O - file error has occurred.
P - page break (reached max print lines) has occurred.
Q - quit program control code is entered.

R - record search (of any type) program control code is entered (gosub).
S - save program control code is entered. (gosub).
U - up arrow control code is entered.
W - down arrow control is entered.
X - left arrow/backspace is entered.
Y - right arrow is entered.
1 - end key is entered.
3 - page down key is entered.
4 - user defined trap.
5 - user defined trap.
6 - user defined trap.
7 - home key is entered.
9 - page up key is entered.
0 - insert key is entered.
. - delete key is entered.

what to do (GID) - G - goto or gosub, depending on the trap type (P, R, and H are gosubs all of the rest are goto's). I - ignore the event. D - take the default action.

Note: Once a trap is set it will remain the same until it is turned off with an I (Ignore) or until it is reset to the default with a D (Default), or until the program is exited. This is required.

label - This is the line label to which control will be transfer. This is only used with goto and gosub.

TAS EDITOR EXAMPLE:

trap: SAVE GOTO LABEL

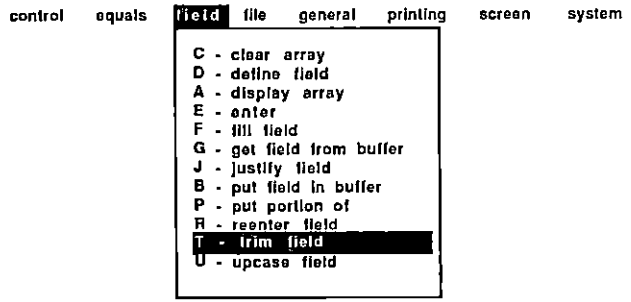
trap: PAGE BREAK GOSUB LABEL

trap: FILE SEARCH DEFAULT

trap: EXIT IGNORE

trap: USER TRAP 1 GOTO LABEL

trim field



The **trim** command is used to trim spaces from an alphanumeric field by converting existing spaces to null characters. This would be used to print two fields right next to each other (i.e. first last name or city state).

PARTS OF THE COMMAND:

trim field - This is the field to be trimmed. It must be an alphanumeric field in the format of *fieldname[ss]*. This is required.

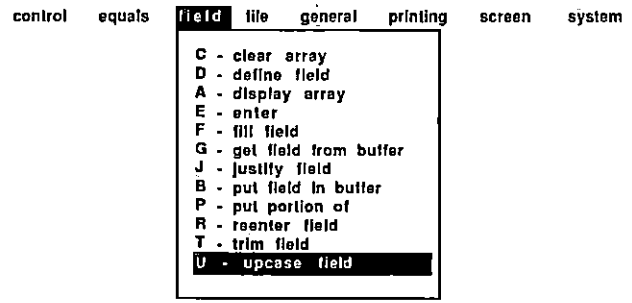
L / T - This tells whether to trim the leading (L) or trailing (T) spaces. This is required.

TAS EDITOR EXAMPLE:

trim blank chrs: TEST LEADING

trim blank chrs: TEST TRAILING

upcase field



The **upcase field** command is used to force all of the characters or part of the charachters in a field to be upper case.

PARTS OF THE COMMAND:

field - This is the field to be checked. It is an alphanumeric in the format of *fieldname[ss]*.

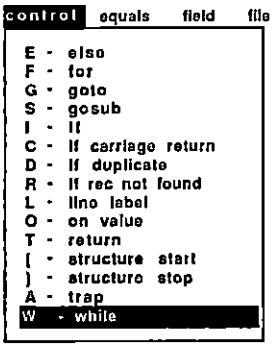
A / Q - A - force all characters to upper case. Q - force only those outside quotes to uppercase.

TAS EDITOR EXAMPLE:

up case: TEST OUTSIDE QUOTES ONLY

up case: TEST ALL

while



The **while** command is used to create a loop that will be executed as long as a specific condition is true. It combines features of the **for/next** command and the **if** command.

The **while** command is like **for** in that it is a loop that can be re-executed several times. However, like the **if** command, **while** must be evaluated as true before the commands in the loop will be run. Unlike the **for** command, if it is evaluated as false, then the loop will not be executed even once.

The line immediately after the **while** command should be a **structure start**. If the command is evaluated as true each command line will be executed until the matching **structure stop** is found. At this point the program will return to the **while** command line and it will be evaluated again. This will continue until the command is evaluated as false, then the program will jump to the end of the command group and continue executing the rest of the program.

PARTS OF THE COMMAND:

compared field - This is any legal field name in the format of *fieldname[ss]*. It may be

alphanumeric, numeric, date, or if a defined field it may be record or overlay type.

how to evaluate expression: These are the comparison types:
< if field is less than expression
<= if field is less than or equal to expression
= if field is equal to expression
=> if field is equal to or greater than expression
> if field is greater than expression
<> if field is not equal to expression

expression - The expression may be a single field, several fields, or a combination of fields and constants. Parts of the expression may be surrounded by parentheses to force the program to evaluate those portions first. The first field to be evaluated in the expression will determine whether the expression is treated as alpha-numeric or numeric type. For example, if the first field is " (blank but is defined as alphanumeric) and if a numeric field follows, then it will be added to the end of the alphanumeric field as though it were also alphanumeric:

("+100+200) -> 100200

If the expression is determined to be alphanumeric then all fields are added (concatenated) together. If the expression is numeric then the legal operators are addition (+), subtraction (-), multiplication (*) and division (/).

NOTE: Even though it is not part of the command there is an implied **do** at the end of the command line similar in effect to that in an **if** command.

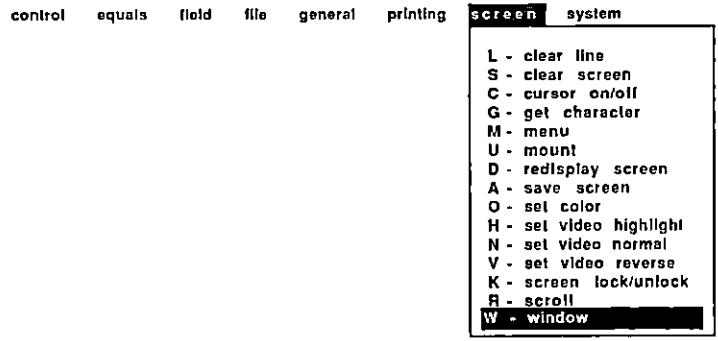
TAS EDITOR EXAMPLE:

```
while: TEST <> (FIELD)

while: TEST < (10)

while: TEST = (FIELD + 2)
```

window



The **window** command is used to create a window (or box) on the screen. It clears the area within the box while leaving all the area outside the box alone. This command can be used to put information on the screen without creating an entirely new screen (or getting rid of the information currently on the screen). It is used in conjunction with the **save screen** command and the **redisplay screen** command. Any number of windows may be open but a maximum of 10 windows can be saved and redisplayed at once.

The upper left hand corner of the screen (location 1,1) becomes the upper left hand corner of the window. The color of the window may be set by using the **color** command immediately before opeing the window but after using the **save screen** command.

Typical usage would be to first save your screen to a buffer with the **save screen** command. Then use the **window** command to open a temporary work area on screen. Then bring back your original screen with the **redisplay screen** command. This happens very quickly on screen.

PARTS OF THE COMMAND:

Lower left column value - This is the column position of the lower left hand corner. It is a numeric constant or numeric field using the standard format of *fieldname[ss].*

Lower left row value - This is the row position of the lower left hand corner. It is a numeric constant or numeric field using the standard format of *fieldname[ss].*

Height value - This is the number of row high that the window will be. It is a numeric constant or numeric field using the standard format of *fieldname[ss].*

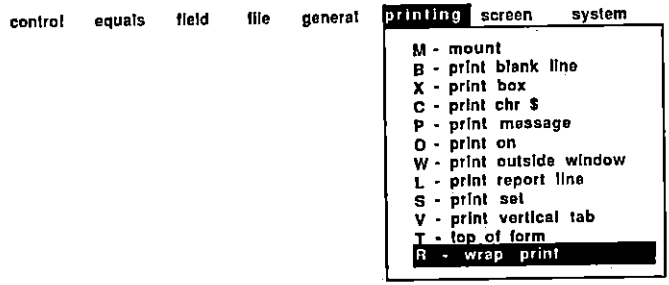
Width value - This is the number of columns wided that the window will be. It is a numeric constant or numeric field using the standard format of *fieldname[ss].*

TAS EDITOR EXAMPLE:

TAS window: column: 10 row: 20 height: 15 width: 45

TAS window: column: COLFIELD row: ROWFIELD height: HTHFIELD width: WIDFIELD

wrap print



The **wrap print** command is used to display lines of characters on screen that exceed the width of the screen or active window. When the line of characters reaches the right side of the screen or active window, it will automatically wrap to the next line at whatever column is specified.

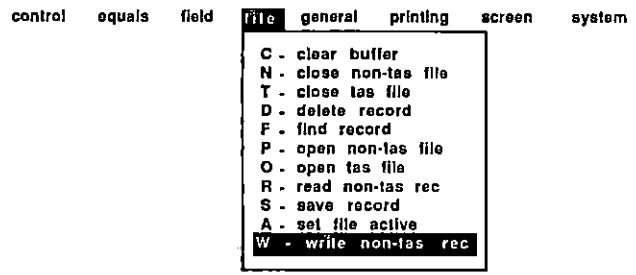
PARTS OF THE COMMAND:

print coninued line at - This is the column number that the text will begin at in the next line. The command may be turned off by setting this value to 0.

TAS EDITOR EXAMPLE:

force line wrap: print continued line at: 10

write non-tas rec



The **write to non-tas file** command is used to write a record to a foreign file from the buffer specified in the **read non-tas rec** command. It always writes sequentially starting with record number 1.

PARTS OF THE COMMAND:

on error goto label - This is where program control will be transferred if a file error occurs during the write.

TAS EDITOR EXAMPLE:

write record in non-tas file on error goto: LABEL

VIII. Expressions: Operators and Hierarchy

Expressions are calculations or formulas. They may consist of only a fieldname or a constant when you are setting the value a field, or they may be a mathematical computation.

Valid operators for all expressions are add (+), subtract (-) multiply (*) and divide (/). Additional valid operators for If and While expressions are less than (<), greater then (>), equal to (=), less than or equal to (<=) and equal to or greater than (=>).

Numeric and alphanumeric constants follow the same rules as explained in Reference Chapter I.

The hierarchy for the order in which calculations are performed is inside out and left to right. This means that operations inside parentheses are done first, and then the formula is calculated from left to right. You may have parentheses nested up to 10 levels deep (parentheses inside parentheses).

Example:

NUMBER = (10-1*(2*4)) Expression equals 72

IX. How to Convert Data Types

Fields of different types can be easily converted in TAS-Plus using the Equals command. When converting from alphanumeric to numeric, the resulting number will be properly right justified and will have the decimal precision of the receiving field. If the alphanumeric field has non-numeric values the resulting number will be unreliable.

When converting from a numeric value to an alphanumeric field the result will be right justified. Decimal characters will be transferred if they exist. For example, if a numeric field defined as 5 characters long with 2 decimal characters has the value of 101, an alphanumeric receiving field with a length of 10 would have a value of " 101". However, if the value of the numeric field is 101.10 then the result is " 101.1".

Dates and times are stored as numeric values to facilitate mathematics with these fields. Therefore, if you set an alphanumeric field equal to a date or time field, you will see the numeric representation instead of the normal alphanumeric representation. For example, if a date field has a value of 03/30/86 and an alphanumeric field is set equal to it, then the result will be "2446520". To get the date in the proper format, add it to a null ("") string. The null string does not add any spaces of its own but does alert TAS-Plus that you want the date (or time) in a normal representation instead of a number.

Example:

ALPHA equals: " "+DATE.FLD

To set the value of a date or time field you may use either a number or an alphanumeric field or constant.

Examples:

DATE.FLD equals: "03/30/86"
DATE.FLD equals: ALPHA

The program will make the proper conversions. This allows you to transfer data from other sources and still utilize the advanced TAS-Plus date and time arithmetic features.

You may also use the Equals portion of command on a date or time field without first converting to an alphanumeric field. The program will do the conversion internally and and treat it as though it were an alphanumeric field.

Example:

ALPHA equals portion of: DATE.FLD start: 1 numchrs: 2

This command line will take the first two characters of the date (month) and place them in the field ALPHA.

ALPHA equals portion of: TIME.FLD start: 4 num chrs:2

This command line will take the minutes characters and place them in the field ALPHA.

X. List of Error Messages

COMPILER ERROR MESSAGES

Programs generated by TAS-Plus through the Create program and Create report selections from the Main Menu will not have compiling errors. When you create or modify programs using the Edit program Main Menu selection, you may introduce errors. You can look up the error message on the alphabetical list below to determine the cause and remedy.

You can produce a program listing with line numbers through the Print program selection on the Activity menu of the source code editor. The program listing will show two sets of line numbers on the left side. The first set is the actual line number. This is the line number that the compiler error message refers to.

ARRAY NUMBER CONSTANT OVER 254

An array element constant of 255 or greater was found in the program. The maximum array element allowed is 254. Replace the constant with a valid constant.

ARRAY SPECIFIED IS OUT OF LIMITS

An array element was found in the program that is beyond the number of array elements defined for the field. Check to see if the element number was typed correctly or increase the number of elements defined for the field.

COMMAND FILE ERROR

An error occurred while creating the compiled program. Check to see if there is enough room on the disk drive and directory you specified.

DATA DICTIONARY RECORD NOT FOUND

An error occurred while reading the data dictionary file. Reindex the data dictionary (TASDICT).

DATA DICTIONARY NOT FOUND

The Data Dictionary master or key file was not found. They must be on the current disk drive and directory in order to compile. Copy the missing file onto the current work area. The filenames are TASDICT.M and TASDICT.K.

DUPLICATE LABELS

A duplicate line label was found. Each label must be unique. Only the first 10 characters of a label are used for identification, so make sure there is no duplication within the first 10 characters.

ERROR WHEN CREATING COMMAND FILE

An error occurred while creating the compiled program. Check to see if there is enough room on the disk drive and directory you specified.

FIELD IS NOT A KEY

A fieldname was found used in a situation where it is required that the field be a key field. Most commonly, a non-key field was used in a Find record command. You can change the field to be a key using the Maintain data dictionary selection on the Main Menu and then reindex the datafile. Or, you can change the field specified in the Find record command to one that is already a key.

FIELD MUST BE OF ALPHANUMERIC TYPE

A command was found using an inappropriate field type. Use of field of alphanumeric type instead.

FIELD MUST BE OF NUMERIC TYPE

A command was found using an inappropriate field type. Use of field of numeric type instead.

FIELD MUST BE OF NUMERIC TYPE AND 8 CHRS

A field of the wrong structure was found in a define command for receiving a record number type. A temporary field for storing a record number type must be of numeric type and exactly 8 characters long. Redefine the field to the proper structure.

FIELD MUST BE OF DATE TYPE

A command was found that requires a field defined as D (long date) or S (short date). Use a field of the correct type.

FIELD MUST BE OF TIME TYPE

A command was found that requires a field defined as T (time). Use a field of the correct type.

FILE LOCATION HOLDER NOT FOUND

The locator master or key file was not found. They must be on the current disk drive and directory in order to compile. If you are using the multiple companies option, there must be a locator master and key file for each company code specified in the program. Copy the missing file onto the current work area. The filenames are TASFILE.M and TASFILE.K.

FILE NOT FOUND

The filename was not found in the locator file. Enter the filename in the locator file (TASFILE) using the Maintain locator file selection on the Main Menu.

FILE NOT IN LOCATION FILE

A record for a datafile was not found in the locator file. Use the Maintain locator file selection from the Main Menu to add one.

FLD NAME OVER 15 CHR

A field name over 15 characters long was found. Reduce the length.

FLD NOT FOUND IN DD

A fieldname you specified in the program was not found in the data dictionary. Check to see if the spelling of the fieldname is correct. Sometimes this is caused by a constant not having quotes around it. Please see the section on conventions.

LABEL NOT FOUND

A line label referred to in a command was not found in the program. Check to see if the line label exists and if it is spelled correctly.

KEY READ ERROR

A problem was encountered while reading the dictionary index or locator file index. Use the Reindex database selection on the Main Menu to reindex the datafiles TASDICT and TASFILE.

ONLY CONSTANT ARRAY NUMBERS ARE ALLOWED ON A SCREEN

A screen or report format was found with an array field using a subscript. Array elements on screens and reports must have constants for the array number.

Example: valid reference - arr.fld[23] invalid reference - arr.fld[num]

PARENTHESES ARE UNBALANCED IN EXPRESSION

An unbalanced parenthesis was found in an expression. There must be an equal number of open and close parentheses in an expression. Add the missing parenthesis or remove the extra one.

SCREEN NOT FOUND

A screen or report format specified in a Mount command was not found. It is either missing or the name in the Mount command was typed differently than the name given in the screen painter.

SOURCE CODE FILE NOT FOUND

The source file specified was not found. Check to see if 1) you typed the name correctly, 2) the file is present on the disk drive and directory you specified and 3) the file ends with a .EDT or .SRC extension.

THE BRACES ARE UNBALANCED

An uneven number of open and close braces were found. There must be an open brace for every close brace. Add the missing brace or remove the extra one.

TOO MANY COMMAND LINES

The program is too long. Programs may be up to 4,500 lines long. Reduce the number of lines in the program.

TOO MANY DIFFERENT FIELDS HAVE BEEN SPECIFIED

Too many different fields have been used. Programs can name up to 255 fields. A named field is one that is mentioned specifically in the program or on a screen or report format. An array field counts as just one field. Reduce the number of named fields.

TOO MANY DIFFERENT FILES HAVE BEEN SPECIFIED

Too many Open TAS file commands were found. You may open up to 16 TAS data files in a program. Closing some of the files in a program does not allow more than 16 to be opened. Reduce the number of files opened.

TOO MANY DIFFERENT GROUPS WITHIN PARENTHESES

An expression was found with too many nested parentheses. An expression may have up to 10 sets of nested parentheses. Remove some groups of parentheses, apostrophes or break the expression into two steps.apostrophes

TOO MANY DIFFERENT SCHEMAS HAVE BEEN SPECIFIED

Too many Open TAS file commands were found. You may open up to 16 TAS data files in a program. Closing some of the files in a program does not allow more than 16 to be opened. Reduce the number of files opened.

TOO MANY LABELS

Too many different line labels were used in the program. The maximum number of line labels allowed in one program is 500. Reduce the number of line labels in the program.

TOO MANY SCREENS

Too many different screen or report formats have been specified. Each program can have up to 16 screen and report formats. Reduce the number of screen and/or report formats.

TOO MANY VARIABLES TO FIT MEMORY

Too many different fields have been used. Programs can name up to 255 fields. A named field is one that is mentioned specifically in the program or on a screen or report format. An array field counts as just one field. Reduce the number of named fields.

UNBALANCED QUOTE MARKS IN THE COMMAND LINE

A command line was found with an uneven number of quote marks. Single quote marks (apostophes) and double quote marks (quotation marks) are interchangeable in an expression. Apostrophes may not be used in a constant. Constants must have a quote mark at the beginning and at the end. Check the command line for format.

YOU HAVE PLACED A CLOSE BRACE BEFORE AN OPEN ONE

A close brace was found without a preceding open brace. Check to see if you reversed the order of two braces or if you have an extra set.

RUN-TIME ERROR MESSAGES

Sometimes an error will occur while running a TAS-Plus program. A list of error messages appears below.

You can produce a program listing with line numbers through the Print program selection on the Activity menu of the source code editor. The listing will help you determine what is causing the error. The program listing will show two sets of line numbers on the left side. The second set is the Run Line. Since TAS-Plus optimizes the source code during compiling, some of the lines are skipped when the actual run-time program is generated. This is the line number that the run-time error message refers to.

ADDING AN ALPHA FIELD IN A NUMERIC EXPRESSION

An attempt was made to add an alphanumeric field to a numeric field. You can do the reverse (add a numeric field to an alphanumeric field) and the whole expression will be converted to alphanumeric. You can get the value of a single alphanumeric field in a numeric field, as long as the value is made up of numeric digits with an optional negative sign and decimal point. Please see Reference Chapter IX for more information on converting data types.

ARRAY ELEMENT 0 NOT ALLOWED

An array field was addressed using a subscript of 0. It might be a field with a value of 0 at the point in the program it is being used as a subscript. You can use the PRINT MESSAGE command to print the value of the field just prior to it being used as an array element number to check this. The range of valid array element numbers starts with 1.

ARRAY ELEMENT OUT OF LIMITS

The number used for the subscript of the array element is not in the range specified in the data dictionary or program DEFINE command. It might be a field with a value out of range at the point in the program it is being used as a subscript. You can use the PRINT MESSAGE command to print the value of the field just prior to it being used as an array element number to check this. Use a number in the range or increase the range. The maximum number of elements that can be defined is 255.

BAD LOCATION

A disk drive that doesn't exist was specified for the location of a program. Check the location drive.

BAD RUN PROGRAM

You are trying to run a program that did not compile successfully. Recompile the program.

BTREE ERROR

An error has occurred while accessing a key file. Reindex the datafile in question.

DICTIONARY KEY FILE NOT FOUND

An operation was tried that requires the data dictionary files to be present. Make sure the files TASDICT.M and TASDICT.K are present on the current drive and directory.

DICTIONARY MST FILE NOT FOUND

An operation was tried that requires the data dictionary files to be present. Make sure the files TASDICT.M and TASDICT.K are present on the current drive and directory.

FIELD NOT FOUND

The field entered was not found in the data dictionary. Check the spelling of the fieldname or enter it in the data dictionary.

FILE NOT ACTIVE

An attempt has been made to access a file, or a field within a file, before it has been opened. This often occurs because a screen or report format has been mounted before the datafile is opened. Check to make sure the OPEN command precedes use of any fields from the file.

FORMAT NUMBER ERROR

A report format line number was used beyond the number of lines present in the report format. Use a valid format number.

KEY FILE NOT FOUND

The key file (.K) was not found for a datafile in the specified location. Make sure that the drive and directory path were entered properly. Also, if you are using the multiple companies option, check to see that there is a .K file with the correct company code. Please see the Multiple Companies section of Tutorial Chapter XII for more information on company codes.

KEY FILE READ ERROR

An error has occurred while trying to read a record from a key file. Reindex the datafile.

KEY FILE WRITE ERROR

An error has occurred while trying to write a record to a key file. Reindex the datafile.

KEY NOT FOUND ON DELETE

A problem was encountered with the key file. Reindex the datafile.

LOCATION FILE KEY FILE NOT FOUND

The locator key file (TASFILE.K) is not present on the current drive and directory. Also, if you are using the multiple companies option, check to see that the file extension includes the correct company code. Please see the Multiple Companies section of Tutorial Chapter XII for more information on company codes.

LOCATION FILE MASTER FILE NOT FOUND

The locator master file (TASFILE.M) is not present on the current drive and directory. Also, if you are using the multiple companies option, check to see that the file extension includes the correct company code. Please see the Multiple Companies section of Tutorial Chapter XII for more information on company codes.

MASTER FILE NOT FOUND

The master file (.M) was not found for a datafile in the specified location. Make sure that the drive and directory path were entered properly. Also, if you are using the multiple companies option, check to see that there is a .M file with the correct company code. Please see the Multiple Companies section of Tutorial Chapter XII for more information on company codes.

MASTER FILE READ ERROR

An error has occurred while trying to read a record from a master datafile. This is somestimes caused by trying to do a FIND NEXT RECORD or FIND PREVIOUS RECORD immediately after the buffer has been emptied by a delete command. Use a FIND BEGINNING, FIND END, FIND MATCH or FIND GENERIC before using the delete command.

You may have a bad datafile. Try reindexing the datafile with a number one or two less then the number stated in the Reindex

datafile program. If that doesn't work it means the problem is in the middle of the datafile, not at the end. Try Reindexing the datafile with smaller numbers until the error goes away. You will lose some data by reindexing with a number less than what is specified originally. You might prefer to reload the master and key files for this datafile from a backup copy instead.

MASTER FILE WRITE ERROR

You may have a bad datafile. Try reindexing the datafile with a number one or two less then the number stated in the Reindex datafile program. If that doesn't work it means the problem is in the middle of the datafile, not at the end. Try Reindexing the datafile with smaller numbers until the error goes away. You will lose some data by reindexing with a number less than what is specified originally. You might prefer to reload the master and key files for this datafile from a backup copy instead.

MATCHING RECORD NOT FOUND

An exact match was not found for the field value specified.

MUST DO SEARCH BEFORE NEXT OR PREVIOUS

A FIND NEXT RECORD or FIND PREVIOUS RECORD command was used when there was no record in the buffer. Use a FIND BEGINNING, FIND END, FIND MATCH or FIND GENERIC first.

NO RECORDS IN FILE

An attempt has been made to read a record from a file that is empty. Put some records in the file.

NOT KEY FIELD

An attempt was made to search on a non-key field. Use a field that is a key field or change the data dictionary to make the field a key field, reindex the datafile (and recompile the program, if there is a program). You may also access datafiles in sequential record number order instead of by using key field searches. See the EQUALS RECORD NUMBER command for more information on this type of access.

PLEASE CHECK YOUR PRINTER - THERE IS A PROBLEM

TAS-Plus is trying to send data to the printer, but isn't successful. Check to see if your printer has power, is plugged into the computer and is on-line.

RECORD NOT ACTIVE

An Attempt was made to delete a record when there was no record in the buffer. Issue a FIND command before trying to delete the record.

RETURN WITHOUT GOSUB

A RETURN command was found with no GOSUB in the logic preceding the RETURN.

RUN ERROR

The program is damaged. Recompile the program.

RUN TIME TOO LARGE

There is not enough memory to load the program you are trying to run. Please see the Technical Specifications in Appendix II for more information.

SCHEMA ENTERED NOT FOUND IN DICTIONARY

A schema was referenced that is not in the data dictionary. Check the spelling of the schema or add it to the data dictionary if it isn't present.

SEARCH REACHED BEGINNING OF FILE

A FIND PREVIOUS command was issued when the first record was already in the buffer. Use a different type of FIND command.

SEARCH REACHED END OF FILE

A FIND END command was issued when the last record was already in the buffer. Use a different type of FIND command.

SPECIAL FILE FIELD SPECIFICATION ERROR

An error has occurred reading fields from a non-TAS file. Check the records for an extra comma or quote, missing comma or quote, or record length beyond the defined limit.

SPECIAL FILE OPEN ERROR

An error occurred while trying to open a non-TAS file. There might not be enough room left on the disk or there might be a write protect tab on the disk.

SPECIAL FILE READ/WRITE ERROR

An error occurred while reading from or writing to a non-TAS datafile. If you are reading from an external file, there might be some bad data in the file. If you are writing to an external file, you might either be out of disk space or have a bad sector.

THE END OF THE PROGRAM WAS EXCEEDED

You have reached or gone beyond the end of the program. Add a RUN, GOTO or similar command at the logical end of the program to transfer control.

THE RUN PROGRAM DOES NOT EXIST

You are trying to run a program that does not exist. Check to make sure you typed the name correctly and that the program is present on the drive and directory path you specified.

THERE ARE NO FIELDS IN THE DATA DICTIONARY

The data disctionary has been damaged. Restore the files TASDICT.M and TASDICT.K from a backup disk or from your original program disk.

TOO BIG

The program is too large to fit in memory. Please see Appendix II for technical specifications.

TOO MANY GOSUBS

There are too many GOSUB commands nested in this program. The maximum number of nested GOSUBS in a program is 10. Reduce the number of nested GOSUBS.

TRYING TO CALCULATE THE VALUE OF AN ALPHA FIELD

An alphanumeric field was used in a situation that requires a numeric field. Convert the field to numeric or use a different field.

TAS-PLUS

APPENDICES

Appendix I.

System Requirements

TAS-Plus requires an IBM-PC/XT/AT or 100% PC-Compatible computer. The computer must be equipped with a monochrome or color monitor, 384K of RAM and at least one floppy disk drive or hard disk. MS-DOS 2.0 or higher is required. A printer and second floppy disk drive (on systems without a hard disk) are optional but recommended.

Appendix II

Technical Specifications

	TAS-Plus	TAS-Plus Developer's Version
Maximum files open at one time:	16	32
Maximum key fields per file:	16	32
Maximum number of line labels:	500	500
Maximum number of screen and report formats:	32	32
Maximum nested gosubs:	10	10
Maximum nested parentheses:	10	10
Maximum nested braces:	10	10
Maximum number of lines in a program: (Typical program size is 200 - 300 lines)	4500	4500
Maximum number of lines loaded into Editor:	1000	2500
Maximum number of named fields in a program: **	255	255
Maximum program size: (Typical program size is 3 - 5 K)	64K	64K
Maximum number of array elements:	255	255
Maximum field size: **	255	255
Maximum number of fields per record:	10,254	10,254

Maximum record size:	10,254	10,254
Maximum number of records:	65,535	16,776,960
Minimum memory: ++	340K	500K

** Named fields are fields actually used in a program. They include fields that are defined in the program, fields used for calculations and fields named on screen and report formats. Array fields count as one named field no matter how many elements they have. The maximum field size of 255 applies to single fields and each element in an array field.

++ The minimum memory required refers to memory accessible to TAS-Plus beyond the operating system. Typically, 384K total memory would be sufficient for TAS-Plus and 640K for TAS-Plus Developer's Version.

Appendix III.

Converting TAS Files to Non-TAS Files

TAS-Plus can export dBase files and three types of ASCII files: fixed-length, delimited and text files. TAS-Plus files are converted to dBase files from the TAS-Plus Main Menu. To convert TAS-Plus files to the various types of ASCII files, write a small program following the examples provided below.

Fixed-length files have fields of equal sizes and records of equal sizes. The records in a fixed-length file may or may not be terminated with a carriage return and line feed. Delimited files have variable length fields separated by commas and each record ends with a carriage return and line feed. Text files have variable length records separated with a carriage return and line feed.

CONVERTING TAS TO FIXED LENGTH FILES

1. First, define outgoing fields as alphanumeric temporary fields to store the data before it is exported. Each of the defined fields should be the exact length that you want it to be in the finished file.

```
define: FLD1 type: A size: 25
define: FLD2 type: A size: 10
define: FLD3 type: A size: 13
```

2. Next, open both the TAS-Plus data file and the ASCII file. In the open non-TAS file command, specify F for fixed-length record type and the first ASCII field defined above as the bufferstart. The record length is the sizes of the fields defined above added together. If your records have a carriage return and line feed, add two to the record size.

```
open: TASRECS
open non-TAS file type: Filename: ASCII.DAT bufferstart: FLD1
   record size: 48
```

3. Find the beginning record in the TAS-Plus file based on the key field you want the ASCII file sorted by.

```
find: BEGINNING using field: TASFLD1
```

4. Enter a line label of START. Equate the ASCII fields to the TAS-Plus fields. Then write the record to the outgoing file, read another TAS-Plus record and loop back to the label START.

TASFLD2 is a long date field and TASFLD3 is a numeric field. To convert each to alphanumeric, add the field to an empty string (also called a null string).

In the Find command, the error label is EXIT. This will cause the program to jump to the EXIT line label when there is an error in reading the file. The error is usually generated by reaching the end of the file.

```
START
 FLD1 = (TASFLD1)
 FLD2 = (" " + TASFLD2)
 FLD3 = (" " + TASFLD3)
 write record in non-TAS file
 find: NEXT in file: TASRECS on error goto: EXIT
 goto: START
```

5. Return to TAS-Plus Main Menu at end of processing.

```
EXIT
 run TAS program: "MM"
```

CONVERTING TAS-PLUS TO DELIMITED

1. Define an ASCII field the length of your TAS-Plus record. Use the Csize (character size) in figuring the length. This buffer is the temporary field that your TAS-Plus fields will be stored in before they are written to the ASCII file.

```
define: FLD1 type: A size: 48
```

2. Open both the TAS-Plus file and the ASCII file. In the Open non-TAS file command, specify D for delimited file type and the ASCII buffer field defined above as the bufferstart. For the record length, use a number larger than the defined size for the bufferstart field.

```
 open: TASRECS
 open non-TAS file type: filename: ASCII.DTA bufferstart: FLD1
record size: 50
```

3. Find the beginning record in the TAS-Plus file. The key field you select will be the field the ASCII datafile is ordered by.

```
find: BEGINNING use field: TASFLD1
```

4. Enter a line label of START. Put the TAS-Plus fields in the buffer. Always start with a blank Put field in buffer command to clear the buffer for the next entry. Then write the record, find the next record and loop back to the line label START.

In the Find command, the error label is EXIT. This will cause the program to jump to the EXIT line label when there is an error in reading the file. The error is usually generated by reaching the end of the file.

```
START
 place field: CLEAR FILE BUFFER
 place field: TASFLD1
 place field: TASFLD2
 place field: TASFLD3
 write record in non-TAS file
 find: NEXT in file: TASRECS on error goto: EXIT
 goto: START
```

5. Return to the TAS-Plus Main Menu.

```
EXIT
 run TAS program: "MM"
```

CONVERTING TAS-PLUS FILES TO TEXT

1. Define an ASCII field the length of your TAS-Plus record. Use the Csize (character size) in figuring the length. This buffer is the temporary field that your TAS-Plus field will be stored in before it is written to the ASCII file.

```
define: FLD1 type: A size: 48
```

2. Open both the TAS-Plus file and the ASCII file. In the Open non-TAS file command, specify T for text file type and the ASCII buffer field defined above as the bufferstart. For the record length, use a number larger than the defined size for the bufferstart field.

```
open: TASRECS
open non-TAS file type: T filename: ASCII.TXT bufferstart: FLD1
record size: 50
```

3. Find the beginning record in the TAS-Plus file.

```
find: BEGINNING use field: TASFLD1
```

4. Enter a line label of START. Put the TAS-Plus field in the buffer. Then write the record, find the next record and loop back to the line label START.

In the Find command, the error label is EXIT. This will cause the program to jump to the EXIT line label when there is an error in reading the file. The error is usually generated by reaching the end of the file.

```
START
 FLD1 = (TASFLD1)
 write record in non-TAS file
 find: NEXT in file: TASRECS on error goto: EXIT
 goto: START
```

5. Return to the TAS-Plus Main Menu.

```
EXIT
 run TAS program: "MM"
```

Appendix IV.

Converting Non-TAS Files to TAS-Plus Files

TAS-Plus can import dBase files and three types of ASCII files: fixed-length, delimited and text files. dBase files are converted to TAS-Plus files from the TAS-Plus Main Menu. To convert the various types of ASCII files to TAS-Plus files, write a small program following the examples provided below..

Fixed-length files have fields of equal sizes and records of equal sizes. The records in a fixed-length file may or may not be terminated with a carriage return and line feed. Delimited files have variable length fields separated by commas and each record ends with a carriage return and line feed. Text files have variable length records separated with a carriage return and line feed.

CONVERTING FIXED-LENGTH RECORDS TO TAS-PLUS

1. Define the TAS-Plus receiving fields in the data dictionary. Define incoming fields as alphanumeric temporary fields.

```
define: FLD1 type: A size: 25
define: FLD2 type: A size: 10
define: FLD3 type: A size: 13
```

2. Open both the TAS-Plus file and the ASCII file. In the Open non_TAS file command, specify F for fixed-length and the first ASCII field defined above as the bufferstart. The record length should be the length of your record, plus two characters if there are carriage returns and line feeds between your records.

```
open: TASRECS
open non-TAS file type: filename: ASCII.DAT
```

3. Enter a line label of START. Read a record, equate the TAS-Plus field names to the ASCII names. Save the record and loop back to the label START.

In the Read non-TAS record command, the error label is EXIT. This will cause the program to jump to the EXIT line label when there is an error in reading the file. The error is usually generated by reaching the end of the file.

```
START
  read a record in non-TAS file on error goto: EXIT
  TASFLD1 = (FLD1)
  TASFLD2 = (FLD2)
TASFLD3 = (FLD3)
save record: TASRECS confirm: N clear buffer: Y
goto: START
```

4. Return to the TAS-Plus Main Menu.

```
EXIT
  run TAS program: "MM"
```

CONVERTING DELIMITED RECORDS TO TAS-PLUS

1. Define the new fields in the Data Dictionary. Define the incoming fields in the program as alphanumeric to use as holding fields before they are converted to TAS-Plus. Make certain that the field sizes you use are the maximum field sizes in the delimited file or unpredictable results could occur.

```
  define: FLD1 type: A size: 25
  define: FLD2 type: A size: 10
  define: FLD3 type: A size: 13
```

2. Open both the TAS-Plus and ASCII files. In the Open non-TAS file command, specify D for delimited file type and the first ASCII field defined above as the bufferstart. The record length to use is the sizes of the defined fields above added together.

```
open: TASRECS
 open non-TAS file type: D filename: ASCII.DTA bufferstart:
FLD1 record size:   50
```

3. Enter a line label of START. Read a record, get the ASCII fields from the buffer and save the record.

```
START
 read record from non-TAS file
 TASFLD1 equals delimited field #: 1
 TASFLD2 equal delimited field #: 2
 TASFLD3 equals delimited field #: 3
 save record: TASRECS confirm: N clear buffer: Y
 goto: START
```

4. Return to the TAS-Plus Main Menu.

```
EXIT
 run TAS program: "MM"
```

CONVERTING TEXT FILES TO TAS-PLUS

1. Define a field in the Data Dictionary as the new TAS-Plus field. Define the incoming ASCII field in the program as alphanumeric to use as a holding field before it is converted to a TAS-Plus record. Make certain that the field size you use is are the maximum record size in the text file or unpredictable results could occur.

```
 define: FLD1 type: A size: 48
```

2. Open both the TAS-Plus and ASCII files. In the Open non-TAS file command, specify T for text file type and the ASCII field defined above as the bufferstart. The record length to use is the size of the defined field above.

```
 open: TASRECS
 open non-TAS file type: T filename: ASCII.TXT bufferstart: FLD1
    record size:   48
```

3. Enter a line label of START. Read a record, get the ASCII field from the buffer and save the record.

```
START
 read record from non-TAS file
 TASFLD1 = (FLD1)
 save record: TASRECS confirm: N clear buffer: Y
 goto: START
```

4. Return to the TAS-Plus Main Menu.

```
EXIT
 run TAS program: "MM"
```

Appendix V.

Adding Highlighting and Color to TAS-Plus

Programs and screens can be visually enhanced by adding highlighting on monochrome monitors and color on color monitors. The **Set color values** command is used to change monochrome highlighting and add color in programming. The **Color** selection on the Screen Painter menu is used to alter highlighting and color on formatted screens.

TAS-Plus has 128 color combinations available. You can select three of these to use at a time. The default color selections are set up in the installation program. They may be changed at any time in your program by using the Set color value command. The values for the three color combinations are entered under normal, highlight and reverse. You may choose any combination for each of the three settings, so you may select reverse video for the normal setting and plain video for the reverse setting. TAS-Plus comes installed with 7 for normal, 15 for highlight and 112 for reverse.

Customers with some monochrome monitors will get unreadable text by changing the settings since the monitors translate colors into different graphics patterns. These users should retain the factory-installed color settings. You might like to experiment though to see the different effects that are produced. A TAS-Plus program called COLOR.RUN that will let you try out differnt highlighting and color combinations is included in your disk set.

The numerical values of the various colors are listed in the table below.

BACKGROUND :	TEXT:
0. Black	0. Black
1. Blue	1. Blue
2. Green	2. Green
3. Cyan	3. Cyan
4. Red	4. Red
5. Magenta	5. Magenta
6. Brown	6. Brown
7. White	7. White
	8. Gray
	9. Bright Blue
	10. Bright Green
	11. Bright Cyan
	12. Bright Red
	13. Bright Magenta
	14. Bright Yellow
	15. Bright White

To choose a color combination, multiply the background color number by 16 and add it to the text color.

White background with red text:
7 times 16 plus 4 = 116

Cyan background with bright blue text:
3 times 16 plus 9 = 57

Black background with bright yellow text:
0 times 16 plus 14 = 14

You can get the text to blink by adding 128 to the number determined for the color combination you want.

Selecting the same background color and print color will make the printing invisible. The TAS-Plus default box-printing character is a double line, but if you choose the same color for printing and background, a thick solid line will be produced. The style of box

drawn may be changed in the installation program. Boxes drawn with both the Print box command and the Bar menu command use the normal setting. Unless you have specified otherwise, screen text uses the normal setting and entry data fields use the reverse setting. The moving bars on bar menus use the normal setting.

Some older color boards display yellow instead of brown. Remember to adjust the brightness of your monitor while trying the color combinations as the intensities of the colors change.

VI. Installing and Customizing TAS-Plus

It is not necessary to install TAS-Plus to run it. However, there are default values that may be changed to suit you. To run the installation program, load TAS-Plus by typing TAS at the operating system prompt and pressing return.

When the TAS-Plus Main Menu appears on the screen, the first option, Run TAS program will be highlighted. Press return to select it. Type TASINSTL as the program to run and press return. If the TAS program TASINSTL is not on the current disk drive and directory, you may precede it with that information. For example, A:TASINSTL or C:\TASDIR\TASINSTL.

CURSOR START AND STOP

With PC-DOS the cursor always blinks. You can adjust the size of the cursor by changing the values for these two options. The cursor size values on a monchrome system run from 0 - 13. The cursor size values on a color system run from 0 - 7 (because there are fewer pixels). If you picture the largest cursor possible as a rectangle the size of a character space on the screen, the top line of the cursor is referred to as position 0 and the bottom line as position as 7 or 13, depending on whether it is a color or monchrome monitor.

You can specify any combination of values for cursor start and cursor stop until you find one you like. The default values are:

Monochrome Color

Start - 12 Start - 6
Stop - 13 Stop - 7

COLOR VALUES

Color values are used for specifying highlighting on monochrome systems and color on color systems. There is a full explanation of using highlighting and color in Appendix XI. Color values may be set as values between 0 and 127. The default color values that TAS-Plus uses when it starts are:

Normal Color Value: 7
Highlighted Color Value: 15
Reverse Color Value: 112

DECIMAL CHARACTER

The default decimal character is a period (.). You may specify a different character, such as comma, if you wish.

TEMPORARY RESTRUCTURE FILE PATH AND NAME

TAS-Plus needs a work area while running the Restructure database selection. The work area needs to be at least as large as the datafile that is being restructured. The drive name, directory path and filename for the temporary file go here. The default is the current drive and directory with a filename of RESTRUC.TMP. You may change the filename to any valid DOS filename that does not conflict with another filename. You may also precede the filename with a drive name and/or directory path.

Examples:

RESTRUC.TMP
C:RESTRUC.TMP
C:\TASDIR\RESTRUC.TMP

BOX TYPE

TAS-Plus uses boxes around bar menus, around windows, in the Box command in the screen painter and the.Print Box command in the source code editor. The default style of box is the double-lined box. You may select another style by typing its number as displayed on the screen.

EXITING AND UPDATING TAS-PLUS

To exit the installation program and update TAS-Plus, just press Esc at any point. The current information will be read from the screen and TAS-Plus will be updated. After pressing Esc, it will take a few moments before you notice disk activity and are returned to the TAS-Plus Main Menu.

The new information will not take effect until TAS-Plus is reloaded. When you return to the Main Menu, please press Esc twice to exit to the system. Then type TAS and press return to reload it.

The installation program may be run as often as you want to change the default specifications.

NOTE TO COLOR USERS: As shipped the TAS-Plus cursor is invisible to color users. Prior to running TAS-Plus, the cursor should be set properly for color:

1. Start with the operating system prompt on the screen.

2. Insert the TAS Working disk you made in the tutorial in drive A if you have a floppy system.

3. Type COLOR and press return.

4. When the operating system prompt reappears, your cursor will be activated in TAS-Plus.

This procedure sets the default cursor start and stop values described above to 6 and 7 respectively.

Appendix VII.

FUNCTION KEYS

F1 - Help

Pressing F1 will activate the help facility of TAS-Plus. If no help is available for the current screen, the bell will sound. If context-sensitive help is available it will display, otherwise the Program Control Screen will appear. Press any key to restore original screen.

You may use the context-sensitive help capability of TAS-Plus in your programming. Please see the Enter field command (Help label) for a description.

F2 - Menu

Pressing F2 will cause a jump within a program, usually to a menu. If the bell sounds when F2 is pressed, it means that F2 has been disabled in that part of the program.

F3 - Clear record

Pressing F3 will clear data from the screen. Place the cursor on a field of the record you want to clear from the buffer and press the F3 key. If data is displayed on the screen from more than one file, the other information will stay on the screen until it is cleared.

F4 - Delete record

A record may be deleted from a file using the F4 key. Place the cursor on a field of the record to delete and press F4. You will be prompted to confirm that the record should be deleted. Press return or Y to confirm the delete.

If you hear a bell when you press F4, the delete function has been deliberately trapped to prevent deletion of records.

F5 - Beginning record

The beginning record in a file can be displayed by pressing the F5 key. Your cursor must be in a key field to access the beginning record. You will get a different beginning record at each key field. If your cursor is in a last name field, for instance, and you press F5, the first record alphabetically by last name will appear. If you move to a date field and press F5, the record with the earliest date will appear.

F6 - Ending record

The last record in a file can be displayed by pressing the F5 key. Your cursor must be in a key field to access the last record. You will get a different ending record at each key field. If your cursor is in a last name field, for instance, and you press F5, the last record alphabetically by last name will appear. If you move to a number field and press F5, the record with the highest number will appear.

F7 - Previous record

Pressing F7 will display the previous record. You must first bring a record up by using F6 or F9 in a key field. F7 will then find the record with the next lower key in the file. F7 may not be used immediately after a delete record command. If you see a message that says "Search reached beginning of file" after pressing F7, it means that you are already at the beginning of the file so there is no previous record. Press any key to continue.

F8 - Next record

Pressing F8 will display the next record. You must first bring a record up by using F5 or F9 in a key field. F8 will then find the record with the next higher key in the file. F8 may not be pressed immediately after a delete record. If you get a message that says "Search reached end of file" after pressing F8, it means you are already at the end of the file so there is no next record. Press any key to continue.

F9 - Find record

You may find a record based on a key field in the record by pressing F9. First, place the cursor in the field that you want to search. Then type the value that you want to search for. The value may be an entire field, like invoice number, or it may be a partial field, like the first three letters of a last name. Press F9 after entering the value to search for. Do not press return first. If a record with a matching value is not found, the next closest record in the file will be found. After using F9, you may use F7 or F8 for previous or next record.

F10 - Save record

Pressing F10 will save the current record in a file. You will be prompted to confirm that you want to save the record. Press return or Y to confirm the save. If you hear a bell when you press F10, it means that the save function has been delibrately trapped tp prevent saving a record at that point in the program.

EDITING CONTROL KEYS

<-	Move cursor left one character
->	Move cursor right one character
^	Move cursor to previous field
v	Move cursor to next field
CTRL-D	Delete character at cursor
Del	Delete character to left of cursor
Ins	Insert character at cursor
CTRL-U	Clear data from field
CTRL-R	Restore cleared data to field
CTRL-Y	User control #1
CTRL-Z	User control #2
CTRL-T	User control #3
Esc	Quit program

PROGRAMABLE KEYS

The following keys can be trapped and programmed to perform specific functions. Please refer to the Trap command in the reference section for further explanation.

Key	Action	Trap
F1	Help key entered	H
F2	Jump within program	J
F3	Clear record	C
F4	Delete record	D
F5/F6	Find beginning/ending record	B
F7/F8	Find next/previous record	N
F9	Find generic record	F
F10	Save record	S
Esc	Interrupt operation	I
Esc	Quit program	Q
^	Up arrow key entered	U
v	Down arrow key entered	W
<-	Left arrow key entered	X
->	Right arrow key entered	Y
End	End key entered	1
PgDn	Page down key entered	3
CTRL-Y	User-defined trap #1	4
CTRL-Z	User-defined trap #2	5
CTRL-T	User-defined trap #3	6
Home	Home key entered	7
PgUp	Page up key is entered	9
Ins	Insert key entered	0
Del	Delete Key entered	-

TAS-PLUS

INDEX

Index

A

Actual line number....310
Add a record....45
Add field on screen....172
Application development environment....18
Array....134,148164,178,191,313,317
ASCII files....183,222,244,262,266,306,329,333
Auto carriage return....140

B

Beginning record....215
Beginning record function key....344
Bell....177
Box....173,250,341
Box drawing....88
Braces....137,291,292
Browse database....60,63,156

C

Calculations....307
Change a record....54
Change/add line....166
Clear array....178
Clear buffer....109,180
Clear line....181
Clear record function key....343
Clear screen....120,182
Close non-TAS file....183
Close TAS file....184
Color....174,252,284,288,289,290,337,341
Color users....342
Company code....185,246

Compile program.....154,169,186
Compiling.....91
Config.sys.....5,9
Constants, alphanumeric.....149
Constants, numeric.....149
Control characters.....252
Convert data types.....308
Convert dbase to TAS.....161
Convert TAS to dbase.....160
Create database.....22,23,155
Create program.....67,72,89,91,153,165
Create report.....92,157,165
Cursor.....340
Cursor on/off.....188

D

Data base management system.....17
Data dictionary.....19,72,159,162
Database.....15,16,55,60,68
Date.....189,204,207,309
Date arithmetic.....141
Decimal character.....341
Define field.....117,190
Del screen.....168
Delete file.....194
Delete line.....167
Delete record.....58,192
Delete record function key.....343
Delimited ASCII records.....244
Delimited files.....330
Delimited records.....334
Devices.....255
Display array.....195
DOS commands.....273

E

Edit program.....67,92,102,116,125,131,153,165

Editing control keys....345
Else....197
Ending record....215
Ending record function key....344
Enter....198
Enter at column number....140
Enter at row number....140
Entry options....139
Equals....117,201
Equals active records....203
Equals day....204
Equals location....205
Equals month....207
Equals portion of....208,236
Equals record num....210
Equals size....212
Equals total records....213
Error messages....310
Error messages, compiler....310
Error messages, run-time....316
Exact match....320
Exact search....215
Exit to DOS....161
Expression....314

F

Field chk/add....169
Fields....149
File manager....17
File names....150
Fill field....214
Find beginning....320
Find beginning of file....53
Find end....320
Find end of file....53
Find generic....320
Find match....320
Find next record....53,320
Find previous record....53,320
Find record....50,52,55,58,107,215,286

Find record function key....345
Fixed length ASCII records....244
Fixed length files....329
Fixed length records....333
For/next....218
Formulas....307
Function keys....343
Function keys, default set up....12
Function keys, programming....346

G

Generic search....215
Get character....220
Get field from buffer....222
Gosub....223
Goto....224
Goto label/line #....169
Graphics....174

H

Help function key....343
Help label....139
Highlight....284,288,337

I

If....225
If carriage return....228
If duplicate....229
If rec not found....231
If, then, else logic....135
Initialize database....158
Initialize file....233
Insert On/Off....167
Installation....340

Installation, floppy disk.....3
Installation, hard disk.....7

J

Jump function key.....343
Jump to location.....172
Justify field.....234

K

Keys.....20,50,70,164
Keys, unique.....229

L

Last record.....215
Length.....212
Limitations, Demo system # of records.....4,10
Line label.....120,150,235
Locator file.....160
Lock type.....246

M

Maintain database.....42,55,155
Mask string.....139
Match search.....215
Memory required.....328
Menu.....116,120,236
Midstring.....208,264
Mount.....240
Multiple companies option.....13,144

N

Next record....215
Next record function key....344
Normal video....252,289

O

On value....121,242
Open non-TAS file....244
Open TAS file....246
Overlay....143,164,190

P

Page break....294
Paper length....259
Password type....131,140
Previous record....215
Previous record function key....344
Print blank lines....249
Print box....250
Print chr$....248
Print length....259
Print message....252
Print on....255
Print outside window....256
Print program....168
Print report line....257
Print screen....173
print set....259
Print vertical tab....261
Print width....259
Printing....13
Program generator....89
Programmable function keys....346
Prompts....148
Put field in buffer....262
Put portion of....264

Q

Quit program....345
Quotation marks....148

R

Read non-TAS record....266
Record locking....246
Redisplay screen....267,303
Reenter field....268
Reindex database....158
Reindex file....272
Related files....113
Related record....215
Relational database....101
Remark....269
Remove field....172
Rename file....270
Report format....171
Report writer....100
Reports....240,257
Restructure data file....158,341
Return....271
Reverse video....141,252,284,290
Run DOS command....153,273
Run line number....316
Run non-TAS program....152,274
Run report....98,157
Run TAS program....91,122,152,276

S

Save program....169
Save record....57,156,180,278,286
Save record function key....345

Save screen.....280,303
Screen.....240,267,280,281, 282,303,314,315
Screen add/chg.....167
Screen lock/unlock.....281
Screen painter.....72,171,175
Screen painter, exit.....174
Scroll.....282
Sequential record access.....210
Set color.....284
Set date.....158
Set file active.....286
Set video highlight.....288
Set video normal.....289
Set video reverse.....290
Show field info.....172
Sorting with indexes.....142
Source code editor.....102,165175
Source code editor, exit.....170
Specifications.....327
Standard screen.....171
Structure start.....291
Structure stop.....292
Subroutines.....223,271
System date.....125
System requirements.....326
System time.....125

T

Text files.....332,335
Time.....293,309
Time arithmetic.....141
Top of form.....294
Trap.....295,346
Trim field.....298

U

Unlocking TAS-Plus.....10
Unlocking your system disk.....6
Up arrow label.....140
Upcase.....140
Upcase field.....299

V

Vertical tab.....261

W

While.....300
Window.....236,256,267,280,303
Wrap print.....305
Write non-TAS rec.....306